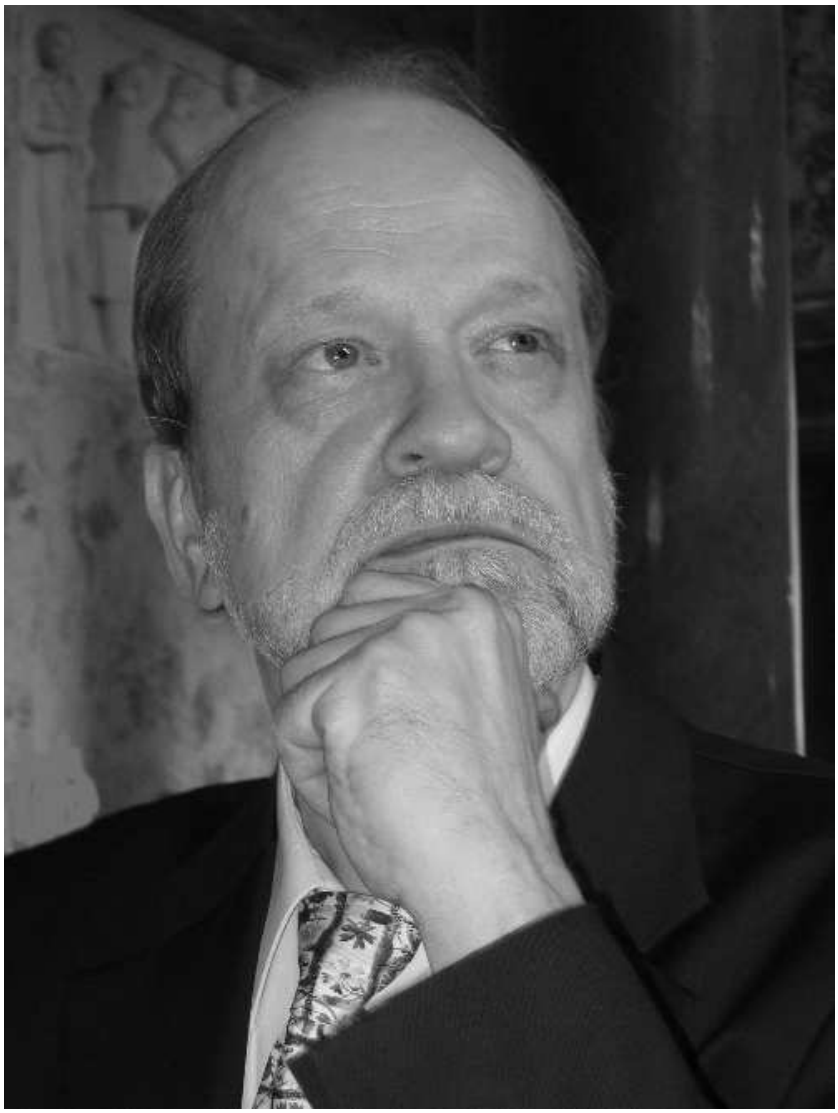
Inquiries into words, constraints and contexts : Festschrift in the honour of Kimmo Koskenniemi on his 60th birthday

Arppe, Antti

CSLI publications
2005

Inquiries into Words, Constraints and Contexts



Inquiries into Words, Constraints and Contexts

Festschrift for Kimmo Koskenniemi on his 60th Birthday

Edited by
Antti Arppe
Lauri Carlson
Krister Lindén
Jussi Piitulainen
Mickael Suominen
Martti Vainio
Hanna Westerlund
Anssi Yli-Jyrä

Technical Editing by
Juho Tupakka

Inquiries into Words, Constraints and Contexts. Festschrift for Kimmo Koskenniemi on his 60th Birthday.

CSLI Studies in Computational Linguistics ONLINE
Series Editor: Ann Copestake
ISSN 1557-5772

Available on-line at:
`http://cslipublications.stanford.edu/site/SCLO.html`

CSLI Publications
Ventura Hall
Stanford University
Stanford, CA 94305

CENTER FOR THE STUDY OF
LANGUAGE AND INFORMATION
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305
(650) 723-1839

CSLI
PUBLICATIONS

Contents

Acknowledgements and Notices

A paper pre-print of this electronic publication was presented to Professor Kimmo Koskenniemi at a special Colloquium on Friday, September 2nd, 2005, at the Auditorium of the Arppeanum building of the University of Helsinki, arranged in association with the Workshop on Finite State Methods for Natural Language Processoring (FSMNLP) <http://www.ling.helsinki.fi/events/FSMNLP2005/>.

This final electronic version of the publication contains a few minor revisions, as compared to the pre-printed paper version. The chapters affected by these revisions are the following: 1, 17, 20, 22, 25, and 26.

The Editorial Board wants to express its appreciation for CSC - Scientific Computing Finland, Lingsoft, Inc., Connexor Oy, and Oy International Business Machines Ab for their generous financial support which greatly contributed to the successful realisation of this project.

Moreover, the Editorial Board expresses its gratitude to Ann Copestake, Dikran Karagueuzian and CSLI Publications for graciously agreeing to publish this Festschrift in their online series.

Furthermore, the Editorial Board wishes to thank Leena Barros and the Office of the Faculty of Arts at the University of Helsinki for administrative support in setting up and running this project.

Finally, the Editorial Board would like to thank the Koskenniemi family for providing the photographs from Kimmo's past and present that are included this Festschrift.

Helsinki, November 11th, 2005,

Antti Arppe	Lauri Carlson	Krister Lindén
Jussi Piitulainen	Mickael Suominen	Juho Tupakka
Martti Vainio	Hanna Westerlund	Anssi Yli-Jyrä

Tabula Gratulatoria

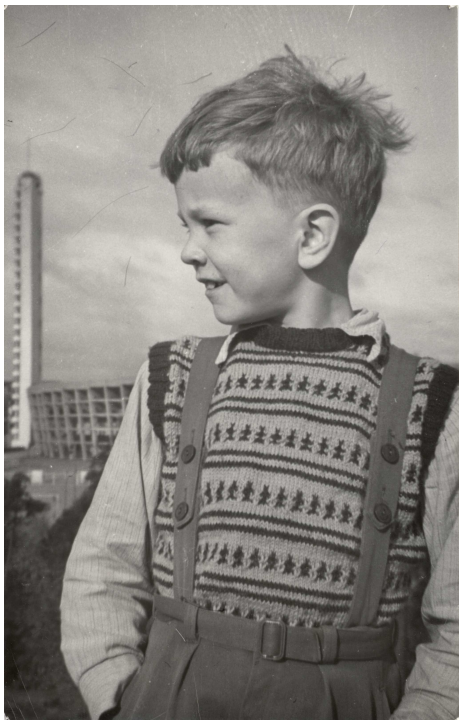
Prof. Olli Aaltonen, University of Turku, Finland
Anders Ahlqvist
Prof. Helena Ahonen-Myka, University of Helsinki, Finland
Anu Airola and Kari Lehtonen
Antti Arppe
Lili Aunimo and Reeta Kuuskoski, University of Helsinki, Finland
Academy Prof. Ralph-Johan Back, Academy of Finland / Department of Computer Science, Åbo Akademi University
Lars Backström
Dr. Kenneth R. Beesley, Xerox Research Centre Europe, France
Dr. phil. Eckhard Bick, Institute of Language and Communication, University of Southern Denmark, Odense
Prof. Lars Borin, Göteborg University, Sweden
Lauri Carlson
Andrew Chesterman, University of Helsinki
Kenneth Ward Church, Microsoft Research, Redmond, USA
Loek Cleophas, Technische Universiteit Eindhoven, The Netherlands
Robin Cooper, Department of Linguistics, Göteborg University, Sweden
Mathias Creutz, Helsinki University of Technology, Finland
Antoine Doucet, University of Helsinki, Finland
Ph.D Steve Farmer
Michiel Frishert
Björn Gambäck, Stockholm, Sweden
Prof. Tapani Harviainen, University of Helsinki, Finland
Orvokki Heinämäki
Prof. Dr. Irmeli Helin, University of Helsinki, Finland

Prof. Dr. Erhard W. Hinrichs, Eberhard-Karls-Universität Tübingen, Germany
Henrik Holmboe, NorFA/NordForsk, Denmark
Prof. Dr. Timo Honkela, Helsinki University of Technology, Finland
Prof. Dr. Arvi Hurskainen, Institute for Asian and African Studies, University of Helsinki, Finland
Sari Hyvärinen, University of Helsinki, Finland
Matti Ihamuotila
Academy Prof. Kalervo Järvelin, University of Tampere, Finland
Prof. Janne Bondi Johannessen, University of Oslo, Norway
Prof. Kristiina Jokinen, University of Helsinki, Finland
Dr. Ronald M. Kaplan, Palo Alto Research Center, USA
Jussi Karlgren, Stockholm, Sweden
Fred Karlsson and Sylvi Soramäki-Karlsson
Prof. Dr. Lauri Karttunen, Stanford University, Palo Alto, USA
Dr Matti Keijola, Helsinki University of Technology, Finland
Prof. Pekka Kilpeläinen, University of Kuopio, Finland
Erkki I. Kolehmainen, Helsinki, Finland
Mare Koit, University of Tartu, Estonia
Jouko and Eeva Koskenniemi
Osmo and Liisa Koskenniemi
Seppo Koskenniemi and Hely von Konov-Koskenniemi
Pirkko Kukkonen
PhD Timo Lahtinen
Ritva Laury
Greger Lindén, University of Helsinki, Finland
Krister Lindén
Prof. Jouko Lindstedt, University of Helsinki, Finland
Mikko Lounela
Urho Määttä
Matti Miestamo
Manne Miettinen, CSC – Scientific Computing Ltd., Finland
Prof. Mehryar Mohri, Courant Institute of Mathematical Sciences, USA
Cand.philol. Sjur Nørstebø Moshagen, Helsinki, Finland / The Sámi Parliament, Norway
Jussi Niemi
Prof. Joakim Nivre, Växjö, Sweden
Attila Novák, MorphoLogic, Hungary

Adjunct Prof. Martti Nyman, Turku, Finland
Prof. Haldur Õim, University of Tartu, Estonia
Marjatta and Asko Parpola, Helsinki
Simo and Sisko Parpola, Helsinki
Jussi Piitulainen
Dr. Ari Pirkola
Helena Pirttisaari, University of Helsinki, Finland
Kari K. Pitkänen
PhD Gábor Prószéky, MorphoLogic, Hungary
CEO Juhani Reiman, Lingsoft, Inc.
Mikael Reuter, Helsingfors, Finland
Prof. Eiríkur Rögnvaldsson, University of Iceland, Reykjavík
Otto-Ville Ronkainen
Klaas Ruppel, Research Institute for the Languages of Finland, Helsinki
Pekka Sammallahti, Ohcejohka, Finland
Kaius Sinnemäki
Koenraad de Smedt and Victoria Rosén, Bergen, Norway
Richard Sproat, University of Illinois at Urbana-Champaign, USA
Dr. Doc. Pirkko Suihkonen, University of Helsinki, Finland
Mickael Suominen
Markku Syrjänen, Helsinki University of Technology, Finland
PhD Pasi Tapanainen, Connexor Oy, Helsinki, Finland
Lauri Tarkkonen
Bertil Tikkanen, University of Helsinki, Finland
Dr. Trond Trosterud, University of Tromsø, Norway
Juho Tupakka
Heli Uibo, University of Tartu, Estonia
Kira and Esko Ukkonen
Prof. Esa Uusipaikka, University of Turku, Finland
Martti Vainio
CDO, Phil. Lic. Simo Vihjanen, Lingsoft, Inc.
Liisa Vilkki, University of Helsinki, Finland
Prof. Dr. Martin Volk, Stockholm University, Sweden
PhD Atro Voutilainen, Connexor Oy, Helsinki, Finland
Prof. Dr. Bruce W. Watson, University of Pretoria, South Africa
Jürgen Wedekind, University of Copenhagen, Denmark
Stefan Werner, University of Joensuu, Finland
Hanna Westerlund
Dr. Graham Wilcock, University of Helsinki, Finland
Anssi Yli-Jyrä

Institutions

Aspekti ry., Helsinki, Finland
Chair of Language Technology, University of Tartu, Estonia
Connexor Oy, Helsinki, Finland
CSC – Scientific Computing Ltd., Finland
Department of Computational Linguistics, Copenhagen Business School, Denmark
Department of Computer Science, University of Helsinki, Finland
Department of General Linguistics, University of Helsinki, Finland
Department of Information Studies, University of Tampere, Finland
GSLT (Graduate School of Language Technology), Sweden
Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic
Oy International Business Machines Ab
Kielikone Oy, Helsinki, Finland
Laboratory of Acoustics and Audio Signal Processing, Helsinki University of Technology, Finland
Lingsoft Inc., Helsinki/Turku, Finland
Research Institute for the Languages of Finland, Helsinki



Kimmo as a child.

The traditional car ride after finishing senior high.

Kimmo Koskenniemi's first 60 years

FRED KARLSSON

Kimmo Matti Koskenniemi was born on September 7, 1945, in the city of Jyväskylä in the interior of Finland, close to those dialect areas where the purest forms of Finnish are claimed to be spoken. Kimmo was the youngest of four sons born to Matti and Sirkku Koskenniemi. Both of his parents were deeply involved with education, his mother as a primary school teacher and his father as one of the leading authorities on education and teacher training in Finland, affiliated as professor of practical pedagogics at the Teacher's College of Jyväskylä 1944-1948 and later at the Universities of Turku and Helsinki.

Kimmo matriculated from one of Finland's top high schools, Helsingin Suomalainen Yhteiskoulu (Helsinki Finnish Co-educational School), and went on to study mathematics and computer science at the University of Helsinki. He was an unusually successful student, obtaining the degree Bachelor of Science in two years and, on top of that, the degree Master of Science in just another year, in 1967.

Kimmo did his military service in the turbulent years 1968-1969, completing his duties in affiliation with the Defence Staff (Pääesikunta), as one of the first officers receiving special training based on their civilian skills (for Kimmo, mathematics and computer science). He was discharged as Second Lieutenant in the spring of 1969.

Kimmo entered working life already in 1966 when he was employed as programmer at the Department of Seismology, University of Helsinki. In the fall of that year he also started teaching, as teaching assistant of mathematics at the Helsinki University of Technology. In 1967 he entered the payrolls of the Computing Centre of the University of Helsinki which was to become

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

his employer for the next 15 years. There he worked as mathematician, senior planning officer, assistant, specialist researcher, and Division Head. Over these years, administrative duties accumulated and gave Kimmo a solid managerial experience even if they also detracted from his scholarly aspirations.

Kimmo's old interest in natural languages grew stronger in the course of the 1970's to the point where he started full-blown academic study of general linguistics, eventually completing his major in this subject in 1979. Since he was a youngster, Kimmo had known the Parpola brothers Asko and Simo, both renowned scholars in ancient languages, and these acquaintances also paved his way to linguistics. Kimmo's first publications from 1979 and 1980 were written jointly with Asko Parpola and treated methodological (including computational) and corpus-linguistic aspects of deciphering the Indus script.

In 1981 Kimmo joined the Department of General Linguistics where he was one of the driving forces in the project Automatic Analysis of Finnish sponsored by the Academy of Finland 1981-1984. If ever a project has turned out an excellent result, this holds of Kimmo's PhD thesis *Two-level morphology: A general computational model for word-form recognition and production* (Department of General Linguistics, Publications No. 11, Helsinki 1983). The degree was conferred upon Kimmo in March, 1984, after a public defence where professor Lauri Karttunen acted as official opponent.

The two-level model TWOL (building on early work by C. Douglas Johnson, Martin Kay, and Ronald M. Kaplan) soon became a classic and the de facto standard of the rapidly evolving field of computational morphology, a position it has retained to this very date. I know of no better way to describe the basic ideas than to cite Lauri Karttunen and Kenneth R. Beesley's "A Short History of Two-level Morphology":

> "Koskenniemi invented a new way to describe phonological alternations in finite-state terms. Instead of cascaded rules with intermediate stages and the computational problems they seemed to lead to, rules could be thought of as statements that directly constrain the surface realization of lexical strings. The rules would not be applied sequentially but in parallel. Each rule would constrain a certain lexical/surface correspondence and the environment in which the correspondence was allowed, required or prohibited. For his 1983 dissertation, Koskenniemi constructed an ingenious implementation of his constraint-based model that did not depend on a rule compiler, composition or any other finite-state algorithm, and he called it TWO-LEVEL MORPHOLOGY. Two-level morphology is based on three ideas: (i) Rules are symbol-to-symbol constraints that are applied in parallel, not sequentially like rewrite rules. (ii) The constraints can refer to the lexical context, to the surface context, or to both contexts at the same time. (iii) Lexical lookup and morphological analysis are performed in tandem." (http://www.ling.helsinki.fi/~koskenni/esslli-2001-karttunen/)

It is extremely uncommon for PhD dissertations to have such a dramatic impact. The search key "two-level morphology" yielded more than 7,000 hits in June, 2005, and the CiteSeer information service lists 109 citations of Kimmo's 1983 dissertation. Evan Antworth's widely used implementation of the two-level model bears the name of the original inventor, giving us PC-KIMMO.

For 20 years now, Kimmo has been a major character on the international scene of computational linguistics and his work was instrumental in launching the Center of Excellence status of the Research Unit for Computational Linguistics (RUCL, 1985-1994) and the Research Unit for Multilingual Language Technology (RUMLAT, 1995-1999), both at the Department of General Linguistics in Helsinki.

Teaching of computational linguistics started incrementally at the University of Helsinki in the late 1980's. Kimmo had been appointed Docent in 1984, and from April, 1990, he has been professor of computational linguistics (with tenure from May, 1992). The 1990's were economically tough in Finland and Kimmo had to invest an enormous amount of work in designing an up-to-date multidisciplinary curriculum for computational linguistics and related disciplines. The full-blown result of this is the nationwide KIT-network (Language Technology Teaching Network) where ten Finnish universities collaborate. This would not have come about without Kimmo's untiring efforts. Closely related to these endeavours are Kimmo's activities in the Nordic Graduate School of Language Technology (where he is Vice-chair) and in the emerging Finnish-Baltic language technology network.

In 1986, Kimmo was one of the two founders of Lingsoft, Inc. where he has acted both as Chief Executive Officer and Chairman of the Board. The success of Lingsoft (which at times had more than 50 employees) is largely due to Kimmo's diligence, foresight, and strategic eye.

Some five years ago Kimmo was offered to become President Elect of the Association of Computational Linguistics. Had he accepted and taken on this duty, he would later have become President of the Association. It is indicative of Kimmo's deep sense of responsibility and determined prioritization that he declined the offer, preferring to devote his energy to developing the KIT network.

As colleague, friend, project leader, and Department Chair, Kimmo is widely known for his smooth manners, his supportive and collaborative attitude, and his meticulous objective scrutiny of whatever problems he is confronted with. The Department of General Linguistics, staff and students alike, and a wide array of colleagues and friends world-wide offer their congratulations to Kimmo on his sixtieth birthday, wishing him many prosperous years to come.

Defending the dissertation.

Left: Opponent Lauri Karttunen. Center: Custos Fred Karlsson.

Publications by Kimmo Koskenniemi

1977

[Editor.] *B6700 käytön opas*. [Helsinki]: Computing Centre, University of Helsinki. 175pp.

1978

Kielitieteellinen tietojenkäsittely LINUS-ohjelmiston avulla. [Helsinki]: [University of Helsinki]. 39pp. *Computing Centre, University of Helsinki. Research Reports, No. 2*. Reprinted: id. 1979.

Suomen kielen sananmuotojen perusmuodon automaattinen päättely: Mahdollisuuksien arviointia ja taivutuksen formalisointi. [Helsinki]: Computing Centre, University of Helsinki. 108pp. *Computing Centre, University of Helsinki. Research Reports, No. 5*.

1979

[Co-author with Asko Parpola.] *Corpus of Texts in the Indus Script*. Helsinki: [University of Helsinki]. 179pp. *Department of Asian and African Studies, University of Helsinki. Research Reports, No. 1*.

1980

[Co-author with Asko Parpola.] *Documentation and Duplicates of the Texts in the Indus Script*. Helsinki: [University of Helsinki]. 98pp. *Department of Asian and African Studies, University of Helsinki. Research Reports, No. 2*.

[Co-author with Ralph-Johan Back.] *Constructing Verifiable Programs: A Design Method and a Case Study*. Helsinki: Computing Centre, University of Helsinki. 65pp. *Computing Centre, University of Helsinki. Research Reports, No. 11*.

1981

"Syntactic methods in the study of Indus script", in Asko Parpola (ed.) *Proceedings of the Nordic South Asia Conference, Held in Helsinki, June 10–12, 1980*, pp. 125–136. Helsinki: The Finnish Oriental Society. *Studia Orientalia,50*.

1982

[Co-author with Asko Parpola.] *A Concordance to the Texts in the Indus Script*. Helsinki: [University of Helsinki]. 201pp. *Department of Asian and African Studies, University of Helsinki. Research Reports, No. 3*.

1983

Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production. Helsinki: University of Helsinki. 160pp. *University of Helsinki, Department of General Linguistics. Publications, No. 11*.

"Two-level model for morphological analysis", in Alan Bundy (ed.) *IJCAI-83. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 8-12 August 1983, Karlsruhe, West Germany*, Volume 2, pp. 683–685. Los Altos, CA: William Kaufmann.

1984

"A general computational model for word-form recognition and production", in *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics, Proceedings of COLING '84, July 2-6, 1984, Stanford University, California, USA*, pp. 178–181. Stanford, CA: Association for Computational Linguistics.

1985

[Co-author with Fred Karlsson.] "A process model of morphology and lexicon". *Folia Linguistica* XIX.1-2, pp. 207–231.

"A general two-level computational model for word-form recognition and production", in Fred Karlsson (ed.) *Computational Morphosyntax: Report on Research 1981–84*, pp. 1–18. Helsinki: University of Helsinki. *University of Helsinki, Department of General Linguistics. Publications, No. 13*.

"An application of the two-level model to Finnish", in Fred Karlsson (ed.) *Computational Morphosyntax: Report on Research 1981–84*, pp. 19–41. Helsinki: University of Helsinki. *University of Helsinki, Department of General Linguistics. Publications, No. 13*.

"A system for generating Finnish inflected word-forms", in Fred Karlsson (ed.) *Computational Morphosyntax: Report on Research 1981–84*, pp. 63–79. Helsinki: University of Helsinki. *University of Helsinki, Department of General Linguistics. Publications, No. 13*.

"FINSTEMS: A module for information retrieval", in Fred Karlsson (ed.) *Computational Morphosyntax: Report on Research 1981–84*, pp. 81–92. Helsinki: University of Helsinki. *University of Helsinki, Department of General Linguistics. Publications, No. 13*.

1986

"Compilation of automata from morphological two-level rules", in Fred Karlsson (ed.) *Papers from the Fifth Scandinavian Conference of Computational Linguistics, Helsinki, December 11–12, 1985*, pp. 143–149. Helsinki: University of Helsinki. *University of Helsinki, Department of General Linguistics. Publications, No.* .

1987

[Co-author with Lauri Karttunen & Ronald Kaplan.] "A compiler for two-level phonological rules", in Mary Dalrymple, Ronald Kaplan, Lauri Karttunen, Kimmo Koskenniemi, Sami Shaio & Michael Wescoat (eds.) *Tools for Morphological Analysis*, pp. 1–61. Palo Alto, CA: Center for the Study of Language and Information. *Report No. CSLI-87-108*.

[Co-editor with Mary Dalrymple, Ronald Kaplan, Lauri Karttunen, Sami Shaio & Michael Wescoat.] *Tools for Morphological Analysis*. Palo Alto, CA: Center for the Study of Language and Information. *Report No. CSLI-87-108*.

"Morphology", in Stuart C. Shapiro (ed.) *Encyclopedia of Artificial Intelligence*, Volume I, p. 619–620. New York, NY, Chichester, Brisbane, Toronto & Singapore: John Wiley & Sons. Second edition: id. 1990.

1988

[Co-author with Laura Kataja.] "Finite-state description of Semitic morphology: A case study of Ancient Accadian", in Dénes Vargha (ed.) *COLING Budapest: Proceedings of the 12th International Conference on Computational Linguistics*, Volume 1, pp. 313–315. Budapest: John von Neumann Society for Computing Sciences.

[Co-author with Kenneth Ward Church.] "Complexity, two-level morphology and Finnish", in Dénes Vargha (ed.) *COLING Budapest: Proceedings of the*

12th International Conference on Computational Linguistics, Volume 1, pp. 335–340. Budapest: John von Neumann Society for Computing Sciences.

1990

"Finite-state parsing and disambiguation", in Hans Karlgren (ed.) *COLING-90. Papers Presented to the 13th International Conference on Computational Linguistics on the occasion of the 25th Anniversary of COLING and the 350th Anniversary of Helsinki University*, Volume 2, pp. 229–232. [Helsinki]: [Yliopistopaino].

[Co-author with Fred Karlsson.] *BETA-ohjelma kielentutkijan apuvälineenä*. Helsinki: Yliopistopaino. 48 pp.

[Book review of] Kenneth W. Church, *Phonological Parsing in Speech Recognition*. Boston, MA: Kluwer Publishers, 1987. *Computational Linguistics* 16.1: 45–46.

1991

"A discovery procedure for two-level phonology", in Antonio Zampolli, Laura Cignoni & Carol Peters (eds.) *Computational Lexicology and Lexicography. A Special Issue Dedicated to Bernard Quemada*, Volume I, pp. 451–465. Pisa: Giardini Editori e stampatori. *Linguistica Computazionale VI*.

1992

[Co-author with Fred Karlsson & Pirkko Kukkonen.] *Kielitieteellisen analyysin harjoituksia*. Helsinki: Yliopistopaino. [2] + 78pp.
Second edition: id., 1993; third edition: id., 1995; fourth edition: id., 1999.

[Co-author with Pasi Tapanainen & Atro Voutilainen.] "Compiling and using finite-state syntactic rules", in Christian Boitet (ed.) *Proceedings of the Fourteenth International Conference on Computational Linguistics, COLING-92, Nantes*, Volume 1, pp. 156–162. Association for Computational Linguistics.

"Computational morphology", in William Bright (ed.) *International Encyclopedia of Linguistics*, Volume 1, pp. 291–293. New York, NY & Oxford: Oxford University Press.
Second edition: id., in William J. Frawley (ed.) *International Encyclopedia of Linguistics*, Volume 1: 377–380. [New York, NY]: Oxford University Press, 2003.

1994

Suositukset tutkimustietokantojen tietosisällöksi ja perustamiseksi: korkeak-

oulujen tutkimustietokantojen seurantaryhmän loppuraportti. Korkeakoulujen tutkimustietokantojen seurantaryhmä, pj. Kimmo Koskenniemi. Helsinki: Opetusministeriö. *Opetusministeriön työryhmien muistioita; 1994, 5*.

1995

[Editor.] *Proceedings of the 10th Nordic Conference of Computational Linguistics, NODALIDA -95, Helsinki 29–30 May 1995*. [Helsinki]: University of Helsinki. [78] pp. *University of Helsinki, Department of General Linguistics. Publications, No. 26*.

1996

"Finite state morphology and information retrieval" *Natural Language Engineering* 2.4: 331–336.

[Co-author with Mariikka Haapalainen.] "GERTWOL – Lingsoft Oy", in Roland Hausser (ed.) *Linguistische Verifikation: Dokumentation zur Ersten Morpholympics 1994*, pp. 121–140. Tübingen: Max Niemeyer Verlag. *Sprache und Information Nr. 34*.

1997

"Representations and finite-state components in natural language", in Emmanuel Roche & Yves Schabes (eds.) *Finite-State Language Processing*, pp. 99–116. Cambridge, MA & London: The MIT Press. *Language, Speech, and Communication*.

1999

Kieliteknologian koulutuksen laajentaminen. Kieliteknologiatyöryhmän pj. Kimmo Koskenniemi. Helsinki: Opetusministeriö. 23pp. *Opetusministeriön työryhmien muistioita*; *23: 1999*.

"Suunnitteilla oleva kieliteknologiaohjelma", in *XXV kielitieteen päivät: Tampereella 15.–16. toukokuuta 1998*, pp. 68–69. [Tampere]: University of Tampere.

2000

"Merkitys tietokonelingvistiikassa", in Anu Airola, Heikki J. Koskinen & Veera Mustonen (eds.) *Merkillinen merkitys*, pp. 115–122. Helsinki: Gaudeamus.

2002

"Is natural language an inconvenience or an opportunity for IR?", in Micheline Beaulieu, Ricardo Baeze-Yates, Sung Hyon Myaeng & Kalervo Järvelin

(eds.) *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, p. 315. New York, NY: ACM Press.

2003

"GENST-NET and beyond", in Henrik Holmboe (ed.) *Nordisk sprogteknologi 2002 – Nordic Language Technology 2002. Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*, pp. 431–436. København: Museum Tusculanums Forlag, Københavns Universitet.

[Co-author with Lauri Karttunen & Gertjan Van Noord] "Finite state methods in natural language processing". *Natural Language Engineering* 9.1: 1–3.

2004

[Co-author with Jaakko Leino, Arne Hedman, Erkki I. Kolehmainen, Klaas Ruppel & Timo Tuhkanen.] *Eurooppalaisen merkistön merkkien suomenkieliset nimet*. [Online publication.] Helsinki: University of Helsinki, Department of General Linguistics. URL: http://www.ling.helsinki.fi/filt/info/mes2/merkkien-nimet.html

"Common documentation of Nordic language technology parties and activities", in Henrik Holmboe (ed.) *Nordisk sprogteknologi 2003 - Nordic Language Technology 2003. Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*, pp. 23–28. København: Museum Tusculanums Forlag, Københavns Universitet.

[Co-author with Robin Cooper.] "Nordic graduate school of language technology", in Henrik Holmboe (ed.) *Nordisk sprogteknologi 2003 - Nordic Language Technology 2003. Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*, pp. 33–40. København: Museum Tusculanums Forlag, Københavns Universitet.

[Co-author with Anssi Yli-Jyrä.] "Compiling contextual restrictions on strings into finite-state automata", in Loek Cleophas & Bruce W. Watson (eds.), *Proceedings of the Eindhoven FASTAR Days 2004, Proceedings*. Eindhoven: Technische Universiteit Eindhoven. 21pp. *Computer Science Reports 04/40*

2005

"Collecting and disseminating information about Nordic language technology in the future", in Henrik Holmboe (ed.) *Nordisk sprogteknologi 2004 - Nordic Language Technology. Årbog for Nordisk Sprogteknologisk Forskn-*

ingsprogram 2000-2004, pp. 21–30. København: Museum Tusculanums Forlag, Københavns Universitet.

"Final report of the PhD training network GENST-NET", in Henrik Holmboe (ed.) *Nordisk sprogteknologi 2004 - Nordic Language Technology. Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*, pp. 63–66. København: Museum Tusculanums Forlag, Københavns Universitet.

Part I

Morphology, Syntax and Automata

1

The Very Long Way from Basic Linguistic Research to Commercially Successful Language Business: the Case of Two-Level Morphology

ANTTI ARPPE

1.1 Introduction

In the Nordic countries, Finland has stood out in the number of start-ups commercializing language technology, as until the late 1990s practically all of the language technology companies founded in the Nordic countries were of Finnish origin. This fledgling Finnish language industry has strong academic roots – a majority of the Finnish IT companies that are primarily involved in creating and providing software products based on language technology can trace their origins to individual researchers or research groups at Finnish universities. But that is where the similarities end. Both in the case of 'older' companies founded in the 1980s and the 'second wave' of the 1990s, the paths and strategies from academic start-ups to commercially functioning corporations have varied substantially. With time, some of these companies have found for themselves clear, profitable niches, but for others the quest still continues. Nevertheless, a major international breakthrough for a Finnish language engineering company is still in waiting. (Arppe 2002)

From research concerning new technology-based startups it is generally known that success is very difficult to predict. A commonly accepted maxim is that out of the twenty start-ups that a venture capitalist invests in, nineteen will at their best barely make even, whereas typically only a single start-up

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

will turn out to be the success story that covers the losses of the others. Even though one can do one's utmost to create an atmosphere which will foster success, one cannot nevertheless control all the external factors in the operative environment, e.g. competitors' actions, national macroeconomic developments, or changes in potential customer expectations, on which the success of any company ultimately hinges. (summarized in Naumanen 2002)

This very same unpredictability and intrinsic riskiness can be said to apply to scientific enterprise. The purpose of this article is to describe how very long, unexpected and winding paths the advancement of science and business can follow by using as a case example the road from basic linguistic research to Kimmo Koskenniemi's dissertation (1983) introducing the two-level model (TWOL), a milestone in computational linguistics, and further on to the final successful commercialization of this model.

1.2 The scientific roots and infant steps of two-level morphology

As Karttunen and Beesley (this volume) outline the individual twists and turns that led to the presentation of two-level morphology by Kimmo Koskenniemi in 1983, these developments will not be discussed in depth in this article. What is worth noting, however, is that the roots of this, in its essence a computational theory are commonly seen to trace back to the sister field of general linguistics, namely to the generative model of the phonological structure of English by Chomsky and Halle in 1968. This seminal work was on its own part a product of a discussion concerning the general modeling of phonetic and phonological structure of any language based on some group of binary distinctive features, initiated by Jakobson, Fant & Halle in 1952.

In conjunction with presenting his theoretical model, Koskenniemi also demonstrated that his approach worked in practice for at least one natural language by implementing the model for Finnish, which was the origin of a software program that was to be later commonly known as FINTWOL. However, there were theoretical doubts as to the general applicability, efficiency and robustness of the two-level model for any given language (Ritchie et al. 1992: 13-39). For instance, Barton (1986) demonstrated that a linguistic description according to the two-level model could in its worst case turn out to be NP-hard. In response to this critique, Koskenniemi and Church (1988) argued that *natural* languages did not exhibit the types of complexity or long-range dependencies which would lead to such computational complexity. For instance, the number of dependencies which were simultaneously in effect over the entire length of orthographical words (and which would thus be a cause of complexity in two-level models) was at its maximum two in natural languages, say vowel harmony in Turkish.

Despite this on-going theoretical debate, researchers started quite rapidly

after Koskenniemi's dissertation to apply the two-level model with varying degrees of comprehensiveness for the morphological description of different languages. At the University of Helsinki alone, Finnish was followed by two-level models for Swedish (Blåberg 1984), Swahili (developed in 1985-89 [1], documented in Hurskainen 1992), Akkadian (Kataja & Koskenniemi 1988), French (Norberg 1988), Russian (developed in 1988-1990 [2], documented in Vilkki 1997, 2005), and English (developed in 1989-1990 [3], documented in Heikkilä 1991 and Smeaton, Voutilainen & Sheridan 1990). [4] In this manner, too, the two-level model was demonstrated to work in practice for a wide range of typologically divergent languages with respect to their morphology, whether these languages were predominantly suffixing, prefixing or infixing, or agglutinative or flexional, or long dead or fully alive.

1.3 The commercial potential of the two-level model

The two-level model had obvious practical uses which had great commercial potential in conjunction with software programs for text processing and storage, especially for any European language other than English. The morphology of contemporary English is close to non-existent, and compound words are not written together. Therefore, in the case of English text the major challenge in developing a spell-checker for a word processor is how to compile and compress a comprehensive list of words in the vocabulary, as neologisms are constructed or introduced via borrowing, not only from Latin, Greek and French, but from practically any language of the world that happens to have a suitable word, e.g. *ombudsman* from Swedish and *sauna* from Finnish. Whatever inflection remains can be taken care of with a very limited set of truncation or rewrite rules, e.g. removing the plural marker *-s* from nouns. Likewise, one need not worry extensively on how to cope with inflected forms or how to separate a compound word into its components in the development of search or indexing functionalities for English text data bases.

Contrary to English, most other European languages employ inflection, often likened to performing the function of prepositions in English, though inflection is by no means limited to this grammatical function. It is essential to understand that inflection is more than just a matter of adding one affix after another to base forms, as the root lexeme and the morphemes adjoined to it, in their theoretical, idealized forms, interact with each other, so that the orthographical surface form, i.e. the spelling of an individual root or a morpheme

[1] Personal communication 11.4.2005 from Arvi Hurskainen

[2] Personal communication 11.4.2005 from Liisa Vilkki

[3] Personal communication 11.4.2005 from Atro Voutilainen

[4] N.B. Many of the two-levels models mentioned here have been substantially developed further since these initial versions and their documentation., i.e. the ones for Finnish, Swahili, Russian and English.

always depends on the entire morphological structure of an inflected word. In further contrast to English, common neologisms in many other European languages are to a great extent constructed by using productive mechanisms such as combining existing words in the vocabulary or by derivation, rather than by borrowing. Therefore, in order to perform spell-checking or indexing in inflecting languages, truncation is simply of no practical use. The example below providing several morphological constructions based on the Finnish word *vesi* 'water' illustrates this perfectly: [5]

vesi+SG(Singular)+NOM(Nominative):vesi 'water'
vesi+SG+GEN(Genitive):veden 'of water'
vesi+SG+ESS(Essive):vetenä 'as water'
vesi+DN-NEN(Nominal Derivation with -*nen*):vetinen 'watery'
vesi+DN-STO+SG+NOM:vesistö 'water system (group of waters)'
vesi+SG+NOM#pula+SG+NOM:vesipula 'shortage of water'
vesi+SG+GEN#tarve+SG+NOM:vedentarve 'need of water'
vuoro+SG+NOM#vesi+SG+NOM:vuorovesi 'tide (water)'

Thus, to quantify the nature and magnitude of the challenge faced in developing language tools for languages other than English, in e.g. Finnish one can theoretically in the case of the open word classes construct some 2,000 different inflected forms for every noun, 6,000 for every adjective, and 20,000 for every verb. [6]

Should one want to enumerate all the possible inflected forms of, say the 100,000 most common and frequent Finnish words of these open (inflecting) word classes, assuming a distribution as observed in newspaper text, [7] the theoretical sum total would exceed well over 300 million word forms[8]. Even

[5]Notation: MORPHOLOGICAL STRUCTURE:SURFACE FORM; where '+' denotes a morpheme boundary, and '#' denotes a compound boundary

[6]The exact number of morphologically constructible forms is often calculated as 1,872 for Finnish nouns (2 numbers X 13 cases X 6 possessives X 12 clitics) and over 20,000 for Finnish verbs, the latter figure depending on how participle forms are counted in the figure ([530 finite forms + 320 infinitives] X 12 clitics + 5 participles X 1,872). The number of so-called core forms, ignoring clitics, is considerably smaller. Of all of these forms, only a fraction can be observed in even very large corpora of millions of words (personal observations of the author in context of this and earlier work)

[7]This distribution is based on two month's worth of Helsingin Sanomat, Finland's major daily newspaper (January and February 1995), available at the Finnish text bank (Helsingin Sanomat 1995) and automatically morphologically analyzed with the Functional Dependency parser for Finnish (FI-FDG) developed by Connexor (Tapanainen & Järvinen 1997). Selecting the base forms in this corpus of approximately 3.2 million unambiguously analyzed running words, all these different inflected forms were found to represent 113,626 common or proper nouns, 12,005 adjectives and 6,641 verbs. On the basis of this, a rough distribution into 86% nouns, 9% adjectives and 5% verbs was established.

[8]86,000 nouns X 2,000 + 9,000 adjectives X 6,000 + 5,000 verbs X 20,000 = 172M+54M+

with the best compression algorithms there would be no point in trying to generate and list all these forms.

A morphological analyzer program developed according to the two-level model can provide the base form for any inflected word, as long as this word can be constructed using the root lexemes and the morphological rules concerning compounding, derivation and inflection. With these same prerequisites, such an analyzer can also provide the components of any compound word. Furthermore, if a word can be analyzed, provided that the incorporated model is an accurate representation of the orthographical and morphological rules and norms of a language, this will mean that such a word is correctly spelled – in the language in question, that is. Therefore, a two-level model can be used as a basis for the significant improvement of spell-checking and indexing tools for languages with extensive inflection, derivation or compounding. In the case of spell-checking, one needs only to include the root lexeme and its inflectional category in the lexicon in order to recognize not only all the inflected and derived forms of the root but also all the compound words in which it might be used. In the case of indexing and search, one can accurately retrieve all the occurrences of both the inflected forms of a base form and its occurrences as a component of compound words, which is many cases would have been practically impossible or useless with the use of truncation or wildcards. For example, with the two-level model for Finnish, the rather complex but actually observed compound word *väitöskirjatyönohjausajanvarauslista*, i.e.

väittää+DV-OS+SG+NOM#kirja+SG+NOM#...
...työ+SG+GEN#ohjata+DV-US+SG+NOM#...
...aika+SG+GEN#varata+DV-US+SG+NOM#lista+SG+NOM

'reservation list of guidance times for dissertation work' can be correctly recognized in all its inflected forms, e.g. *väitöskirjatyönohjausajanvarauslistallanihan* 'surely on my reservation list for ...', and it can be correctly retrieved using any of its components, e.g. *väitöskirja* 'dissertation' or *aika* 'time' The detection of compound boundaries can also be used to improve hyphenation, as some valid hyphenation borders cannot be detected solely according to character-based rules.

Another commercially interesting property of the two-level model lies in the fact that the model can intrinsically be operated in both directions, i.e. in addition to analysis it can also be used to generate any acceptable morphological form or combination of a root lexemes according to the incorporated linguistic rules. In languages with extensive morphology, this feature turns out to be very useful in the generation of suggestions for corrections of misspelled words, as these correct forms cannot be comprehensively enumerated

100M = 326M word forms

due to the reasons presented above.

The very embodiments of this bidirectional nature of the two-level model are so-called inflecting thesauri, which combine the semantic content of a synonym dictionary or thesaurus with a two-level morphological model for the appropriate language. In these linguistic tools, provided an inflected form as input, the analytical capability of the two-level model is first used to retrieve both the base form and the associated morphological data. Then, the base form is used to retrieve the appropriate synonyms. Finally, the generative capability is coupled with the original morphological analysis data to provide the synonyms of the originally input word in the matching morphological forms.

Nevertheless, one must remember that the two-level model is in the first place a morphological, i.e. structural, rather than a semantic model, and was originally used for linguistic analysis and recognition, in which case the input language is assumed to be orthographically correct. Thus, the recognition of a word does not mean that the recognized word is necessarily a good one in the given context or that it semantically makes any sense – it simply means that the word is morphologically possible. All too often the typos of very common words can be given such a theoretically possible but amusing interpretation, e.g. **ko#mission* 'cow mission' instead of *kommission* 'commission', or **vis#te* 'song tee' instead of *visste* 'knew' in Swedish. Likewise, whereas it is very satisfactory both as an end-user and as a developer of a spell-checker to receive *kielitiede* 'linguistics' as the only and correct suggestion for the typo **kielittiede*, this is not the case for a slightly different typo, **kielitide*, where one has to sift through six other alternatives, e.g. *?kieli#taide* 'language/tongue art', *?kieli#tilde* 'tongue tilde', *?kieli#nide* 'language volume', *?kieli#kide* 'language crystal', *?kieli#side* 'language/tongue tie', and *?kieli#tie* 'language road', which are either odd or utterly jibberish.

However, rising above these undesirable side-effects observed in the development of practical linguistic software, the generative side of a two-level model can be seen from the perspective of general linguistic theory as a manifestation of the semantic potential of the morphological system of a language that it describes, and its misgivings demonstrate how little of this space a language actually uses. Restricting this undesirably excessive word form generation, derivation and compounding without crippling the system's openness is the major challenge in developing spell-checkers, and also inflecting thesauri, based on the two-level model. Discussions concerning the practical extent of this inflectional generality as observed in the development of inflecting thesauri for the Scandinavian languages can be found in Arppe et al. 2000 and Arppe 2001.

1.4 The winding path of commercialization

The two-level model for Finnish attracted rapidly the interest of the industry, and Finnish and foreign companies wanted to study, test or use the software for various purposes either as such or desired some form of additional development. This generated a number of commissioned joint projects which employed many researchers at the Department of General Linguistics from time to time. In the 1980s, however, organizing and managing such commercial projects under the auspices of Finnish universities, even at a small scale, was a novel activity, and in contrast to the present there were even less well established forms for it. Furthermore, the general mood in the Finnish academia at the time was that research and business did not mix well, and this was also the conclusion of Koskenniemi and his collaborator, Fred Karlsson, who as head of the Department had to balance the goals and needs of both the basic academic research and teaching activities and the commercial projects at the Department. Therefore, they decided to move these commercial activities to a private company, Lingsoft, which they founded in 1986 (see Knuuttila 2006 for a detailed description and analysis of the views and motivations of the various actors involved at the Department).

For the rest of the 1980s and the early 1990s, this move appears to have had rather an organizational than an economic effect. Researchers who earlier would have worked in the commercial projects at the Department simply continued the same activities at Lingsoft. Koskenniemi took care of the necessary administrative duties as a part-time managing director while continuing as a full-time senior researcher at the Department. The company had no permanent employees nor did it engage in aggressive marketing activities, and people were employed on a case-by-case basis in order to complete some externally commissioned project, or to pursue some research interest of Koskenniemi or Karlsson, which could be financed from the profits of the commercial projects. Sometimes these noncommissioned projects had no direct commercial goal, but would produce resources that would turn out many years afterwards to be of great value by facilitating, accelerating and in some cases simply making possible some later product development efforts. For instance, as two researchers, Katri Olkinuora and Mari Siiroinen, had compiled a synonym dictionary for Finnish in 1989-91, the company did not have to source and license or develop from scratch this resource when it was necessary in order to develop a Finnish inflecting dictionary for Microsoft in 1995-6.

During this initial, project-driven phase of Lingsoft, the annual turnover of the company hovered on the average at just below one hundred thousand euros. However, at the same time the company succeeded in closing some individual deals, which by themselves even exceeded the average annual

turnover. The most important of such deals was the licensing of the Finnish spell-checker and hyphenator to WordPerdfect in 1988, and the Finnish base form indexing as part of the article data base of Helsingin Sanomat, Finland's largest daily newspaper, in 1992.

In 1992, the persevering efforts at the Department to turn computational linguistics into its own independent discipline bore fruit, as the chair of computational linguistics (renamed language technology in 1999) was established permanently at the University of Helsinki, with Koskenniemi as its first holder. It was then that a slow transformation into a commercial company operating in the traditionally understood sense began at Lingsoft, which was marked by the hiring of the first permanent employee, Krister Lindén, as the managing director. The company embarked on its first major technological development project, undertaken entirely by the company, in order to develop a two-level model for German. This project culminated in 1994 in the overall victory of the first German Morpholympics (Hausser 1996), a competition on developing an efficient and comprehensive morphological analyzer for German in which Lingsoft with its GERTWOL (Koskenniemi & Haapalainen 1994) was the only commercial and non-German participant.

However, this victory did not immediately produce economic returns which had been invested in it, as one might have expected based on the size of the German market and the enthusiastic commercial reception experienced earlier with regards to FINTWOL. Other external developments were also presenting rising challenges for the company. Lingsoft's long-standing partner and customer of proofing tools [9], WordPerfect, was increasingly losing market share in its main business of word-processors to Microsoft, which could leverage its dominance in the operating systems market. Microsoft, on the other hand, already had an existing licensing deal for all its proofing tools for all the major European languages with Inso [10], which had transformed from the spin-off software division of the American publisher Houghton Mifflin into the major player in the language industry in the earlier 1990s. Though Inso's proofing tools were in essence word-list-based, following the English model as presented above, Microsoft was apparently under no sufficient customer pressure to change its subcontractor in 1994-5. [11]

[9] The term Proofing Tools has become to denote not only text-verification programs such as spell-checkers and grammar-checkers also hyphenators and thesauri, i.e. synonym dictionaries, mainly as a result of the influence of the major licensors of these tools, firstly WordPerfect and later Microsoft.

[10] The company in question has operated under several company names, first as InfoSoft International Incorporated 1994-1995, then as Inso Corporation 1995-2000, and finally as eBT International 2000-, being liquidated in 2001. In 1998 the company, then as Inso, sold off its entire linguistic tool business, including customer relationships and contracts, to Lernout&Hauspie, itself now also defunct.

[11] Personal communications in October 1994 and 14.11.1995 from Tarja Tiirikainen, Program

Therefore, from Lingsoft's perspective Microsoft seemed to be a lost cause at that time, and the only available path to generate revenues from proofing tools would be to aim directly at each national end-user market. On the other hand, it appeared that the window for proofing tools as independently marketed software was closing, based on the competitor analysis of e.g. Kielikone, another Finnish language technology start-up, which seemed to have been shifting its marketing and development focus away Morfo, its reputed stand-alone spell-checker for Finnish, to electronic dictionaries. Despite some initial distaste to 'annoying squiggly red lines" marking typos, later developments have shown that proofing tools indeed have turned into the deeply embedded and enabling components of other software programs (EUROMAP 1998: 17-20), the functionality and quality of which are only indirectly visible to the end-users of the parent applications such as word-processors. In conjunction, the structure of the supply chain for proofing tools has developed into a true niche-channel supplier model, with Lingsoft and other small language technology companies in the role of niche suppliers, and Microsoft and other international IT giants as the channels (EUROMAP 1998: 47-56).

Though Microsoft was thus not overtly interested in relicensing its proofing tools, it was interested in localizing AnswerWizard, a fusion of a natural language database query system with a help database, and it was keen on having this work undertaken by a company with linguistic technological competence. Lingsoft was obviously such a company, with demonstrated experience in a variety of languages. However, the range of languages offered to Lingsoft were not only those in which the company had previous experience of its own or through partnerships, such as Swedish and Danish, but also languages with which the company had no real previous competence, such as Norwegian, Dutch and Spanish. Nevertheless, Lingsoft succeeded in negotiating in 1995 a deal covering the localization of AnswerWizard for all of the mentioned languages. Even more importantly, Lingsoft also satisfied Microsoft's quality and other requirements, as the project was renewed, with the addition of Finnish, Russian, Czech and Polish, on several occasions until 2000.

Not only was the AnswerWizard project instrumental in providing Lingsoft desperately needed financial stability in 1995-1996, but by demonstrating the company's capability to undertake such a demanding and complex multlilngual project it also put Lingsoft in a favorable position when Microsoft finally, and in fact quite soon, did decide to reconsider its proofing tool licensing relationships. Thus, Lingsoft had the full package of sufficient financing, right contacts, and good track record, in addition to its birthright of state-of-the-art linguistic technology, in order to be selected in 1996 out

manager for proofing tools at Microsoft.

of three competitors as Microsoft's new subcontractor for the Finnish spell-checker, hyphenator and thesaurus, which relationship Lingsoft has retained with Microsoft ever since. It is a tribute to Koskenniemi's linguistic skills to note that the linguistic description of Finnish incorporated by him in FINTWOL was to a very large extent used in its original form in these proofing tools licensed to Microsoft, and this still continues to be very much the case, even after over twenty years of their original inception.

After this suite of Finnish proofing tools, Lingsoft went on to license to Microsoft the Swedish inflecting thesaurus in 1996, the Swedish spell-checker and hyphenator and the Norwegian (bokmål) and Danish inflecting thesauri in 1997, and the German spell-checker, hyphenator and inflecting thesaurus, and the Norwegian (both bokmål and nynorsk) spell-checkers and hyphenators and the Danish spell-checker and hyphenator in 1998. In addition to these proofing tools based essentially on the two-level model, Lingsoft also succeeded in developing in 1997-1998 and licensing to Microsoft a Swedish grammar-checker (Arppe 2000, Birn 2000), the first of its kind, which was based on the Constraint Grammar formalism originally presented by Fred Karlsson (1990), and realized and further developed by the Research Unit for Multilingual Language Technology (RUMLAT) (Karlsson, Voutilainen, Heikkilä & Anttila 1995). This product development process was successfully duplicated for Finnish, Danish and Norwegian (bokmål), and licensed to Microsoft in 2000-2001. In association with these successful contracts, the number of personnel and the turnover of the company started to grow as presented in Table 1.

1.5 Factors which influenced the commercialization process

From the introduction of the two-level model in Koskenniemi's dissertation in 1983 it took over ten years to transform this theory into a steady commercial income flow of over one million euros in 1996, if measured in terms of Lingsoft's annual turnover presented in Table 1. With the benefit of hindsight one can consider whether it would have been possible to significantly accelerate this process of technology transfer and commercialization.

The basic building blocks used by Lingsoft in its proofing tools, e.g. the two-level models for Finnish and Swedish, had been extensively developed by 1990, which is demonstrated by the licensing deal of a Finnish spell-checker to WordPerfect as early as in 1988. The inhibiting factors were essentially technical in nature and intrinsic to the initial development and implementation of the two-level model. At the University of Helsinki, Koskenniemi had at his disposal the best and most advanced computer facilities available to anyone in Finland, which already by the beginning of the 1980s used multiprogramming operating systems with virtual memory. As a component of a

FIGURE 1 Lingsoft's turnover and personnel 1992-2005.

Year	Turnover	Personnel
1992	0.4 MFIM (0.06 M€)	2
1993	1.3 MFIM (0.22 M€)	5
1994	1.4 MFIM (0.23 M€)	6
1995	4 MFIM (0.7 M€)	7
1996	6 MFIM (1.0 M€)	10
1997	8 MFIM (1.3 M€)	16
1998	13 MFIM (2.2 M€)	20
1999	8 MFIM (1.3 M€)	25
2000 (15 months)	17 MFIM (2.8 M€)	30
2001	1.2 M€	60
2002	0.15 M€	10
2003 (Pasanet merger)	0.25 M€	4
2004 (estimate)	0.08 M€	10
2005 (estimate)	1.9 M€	15

functioning computer program, the obvious data structure into which the two-level model could be transformed was a single finite-state automaton, which in the case of the original FINTWOL consumed several hundred kilobytes of memory, competing with the memory needs of other applications. This had not been a problem in the computing facilities at the university. However, in the realm of personal computers which were the source of commercial potential for general software programs, as Microsoft's MS-DOS and Windows operating systems had gained a dominant position by the beginning of 1990s, operating systems that in practice allowed for running multiple applications concurrently, with genuinely flexible and sufficient virtual memory, were to spread broadly on the general consumer market only with the introduction of the Windows95 in 1995. In principle, it would have been possible to hack a solution to make two-level models (of the full-fledged size necessary for e.g. spell-checking) work with the memory constraints of earlier applications and operating systems – certainly the existence of Kielikone's Morfo spell-checker was a practical proof of its feasibility – but with the limited personnel resources at the disposal of the company it was simply not considered worth all the effort, as the inevitable arrival of Windows95 was more or less certain for several years before its eventual launch. [12] At the time, these arguments seemed from the perspective of Lingsoft's marketing personnel some sort of

[12] Personal communications with Pasi Ryhänen in 18.3.2005 and Mikko Silvonen on 18.3.2005, who were both senior software engineers and product development managers at Lingsoft throughout the 1990s.

unfounded resistance or reluctance of the software developers, but now it is quite clear that in 1994 or even in 1993 the necessary development investments would not have been offset by any potential benefits and associated incomes in the year or two by which Windows95 had effectively replaced earlier PC operating systems.

Even without this intrinsic technical restriction on the spread of linguistic tools exploiting two-level models, the embedded nature of proofing tools had the logical consequence that no proofing nor other purely linguistic tools would have had any commercially interesting market potential before the spread of word-processors or text data bases, which has been described as the product adoption hierarchy of linguistic tools (Arppe 1995a, 1995b). Without software programs that enabled the electronic authoring of text there would hardly have been any commercial need for software programs to spell-check such texts. Even more fundamentally, electronic text authoring tools could become general household consumer tools only with spread of personal computers, which started in earnest with the introduction of IBM's first PCs in the early 1980s – at the very same time that the two-level model was conceived of in the first place. Therefore, it is difficult to see how proofing tools which capitalized on the rule-based, open nature of the two-level model, allowing for a crucial improvement when compared with the preceding list-based solutions, could in practice have been successfully commercialized essentially earlier than how the events folded out in practice.

1.6 Conclusions on the nature of scientific and commercial advancement

In conclusion, Koskenniemi's two-level model in 1983 was a practical computational solution to originally linguistic research questions and subsequent discussion which can be traced as far back as 1968, and even earlier. On its own part, the commercialization process of the two-level model was greatly dependent on developments in the external IT business environment. The introduction of the first personal computers in the early 1980s and then the spread of word-processors and text data bases in the late 1980s were obligatory prerequisites for the emergence of a need for linguistic proofing tools. The break-through of such solutions based on the two-level model was further dependent on the large-scale spread of 32-bit operating systems starting in the mid 1990s.

With the help of these external developments and as a result of all the development work at Lingsoft in 1986-2001, the company became Microsoft's subcontractor of proofing tools for all the major Nordic languages and German. Despite severe difficulties experienced by the company in 2001-2004 (see Knuuttila 2006 and Arppe 2002), finally relieved by a merger with

Pasanet, a translation and localization company based in Turku, these contracts have all been renewed again in 2004, and in retrospect proofing tools can clearly be seen to have in practice been for Lingsoft both its core technological competence and its main source of income over its entire existence. Thus, Finnish language technology, based on the two-level model and the constraint grammar formalism, is now used by tens of millions of people in the Nordic and German-speaking countries, which can be considered a major success for the Finnish IT industry, and even more so for the Finnish language technology community. In this, Kimmo Koskenniemi has played a central role.

In order for all this to be possible we can see an overall arch of incremental individual advances in basic research spanning over several decades from the 1950s to the 1990s. It is clear that the present, but by no means final scientific and commercial outcomes could not have been predicted or determined at the outset. The history leading to and proceeding on from the two-level model is an outstanding example of how scientific research can produce significant commercial benefits, when it is allow to proceed in a free and open manner, and is not constrained by any short-term interests whatsoever.

1.7 Acknowledgements

The historical accounts in this article are based on interviews with Kimmo Koskenniemi and others conducted for my Master's thesis, which Kimmo arranged for me to undertake at Lingsoft in 1994-1995. I furthermore gratefully acknowledge the innumerable discussions with past and present employees at Lingsoft who participated in the development of the company's proofing and other tools, whether the subject was corporate strategy or language technology. Specifically, I would like to thank Thomas Bilgram, Juhani Birn, Era Eriksson, Juhani Järvikivi, Risto Kankkunen, Krister Lindén, Anni Koskenniemi, Ilkka Koskenniemi, Timo Koskenniemi, Ari Majorin, Mariikka Majorin, Sjur Nørstebø Moshagen, Markku Norberg, Ari Paavilainen, Otto-Ville Ronkainen, Pasi Ryhänen, Mikko Silvonen, Eero Taipale, Kimmo J. Tykkälä, Eleonoora Vanhanen, Mari Voipio, Fredrik Westerlund, and Malene Würtz, for being part in realizing the potential of what the two-level model and Lingsoft could be. With regards to this article, I am grateful to Martti Vainio and Anssi Yli-Jyrä for providing input concerning the phonological, computational and philosophical roots of the two-level model, and to Krister Lindén for comments on its commercialization. I am also indebted to Jussi Piitulainen and Kimmo Koskenniemi for encouraging me in 1999 to set up of a lecture course, on which the analysis presented here is originally based upon. Finally, I would like to thank Tarja Knuuttila for providing perspective which I believe has led a more balanced account of the events.

References

Arppe, A. 1995a. *The Strategic Opportunities of a Small High-Technology Company on Emerging Markets.* Unpublished Master's Thesis, Institute of Industrial Management, Helsinki University of Technology.

Arppe, A. 1995b. Information Explosion and the Use of Linguistic Tools in Finland. Harakka, T. & Koskela, M. (eds.) *Kieli ja tietokone.* Association Finlandaise de Linguistique Appliquée, Yearbook 54. Jyväskylä.

Arppe, A, Voipio, M. & Würtz, M. 2000. Creating Inflecting Electronic Dictionaries. Lindberg, Carl-Erik & Lund, Steffen Nordahl (eds.) 17th Scandinavian Conference of Linguistics, Nyborg August 20-22, 1998. *Odense Working Papers in Language and Communication* No 19, April 2000, Vol 1. University of Southern Denmark, Odense, Denmark.

Arppe, A. 2000. Developing a Grammar Checker for Swedish. In: Nordgård, T. (ed.) *Proceedings from the 12th Nordiske datalingvistikkdager*, Trondheim, December 9-10, 1999. Department of Linguistics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Arppe, A. 2001. Lärdomar från utveckling av inflektereande synonymordböcker. Gellerstam, Martin et al (eds.) Nordiska studier i Lexikografi 5. Rapport från *'Konferens om lexikografi i Norden'*, Gothenburg May, 26-29, 1999. Skrifter utgivna av Nordiska föreningen för lexikografi (6) i samarbete med Nordiska språkrådet och Meijerbergs institut, Gothenburg, Sweden.

Arppe, A. 2002. Ei yhtä ainoaa polkua - Suomalaisia kokemuksia matkalla kieliteknologisesta tutkimuksesta liiketoimintaan [No single path - Finnish lessons in the commercialisation of language engineering research]. *Puhe ja kieli* 22:1, pp. 37-44. English translation available at: URL: http://www.hltcentral.org/page-969.shtml.

Barton, E. 1986. Computational complexity in two-level morphology. In: *Proceedings of the 24th Conference of the Association for Computational Linguistics*, pp. 53-59.

Birn, J. 2000. Detecting grammar errors with Lingsoft's Swedish grammar-checker. In: Nordgård, T. (ed.) *Proceedings from the 12th Nordiske datalingvistikkdager*, Trondheim, December 9-10, 1999. Department of Linguistics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Blåberg, O. 1984: *Svensk böjningsmorfologi. En tvånivåbeskrivning.* [The inflectional morphology of Swedish. A two-level model description]. Unpublished Master's thesis, Department of General Linguistics, University of Helsinki.

Chomsky, N. & Halle, M. 1968. *The Sound Patterns of English.* New York: Harper and Row.

EUROMAP 1998. *The EUROMAP Report. Challenge & Opportunity for Europe's Information Society*. Language Engineering Sector of the Telematics Applications Programme, DG XIII Telecommunications, information market and exploitation of research, European Commission, Luxembourg.

Hausser, R. 1996. *Linguistische Verifikation. Dokumentation zur Ersten Morpholympics 1994.* Max Niemayer, Tübingen.

Heikkilä, J. 1991. *A Lexicon and Feature System for Automatic Morphological Analysis of English.* Unpublished Master's thesis, Department of General Linguistics and Department of English, University of Helsinki.

Helsingin Sanomat 1995. 22 million words of Finnish newspaper text. Compiled by the Department of General Linguistics, University of Helsinki and CSC Tieteellinen Laskenta Oy. Available at URL: http://www.csc.fi/kielipankki/

Hurskainen, A. 1992. A Two-Level Computer Formalism for the Analysis of Bantu Morphology. An Application to Swahili. *Nordic Journal of African Studies*, Vol 1/1. Available at URL: http://www.njas.helsinki.fi/.

Jakobson, R., Fant, G. & Halle, M. 1952. *Preliminaries to speech analysis: the distinctive features and their correlates.* (MIT Acoustics Laboratory Technical Report 13.) MIT Press, Cambridge, Massachusetts.

Karlsson, F. 1990. Constraint Grammar as a Framework for Parsing Unrestricted Text. In: Karlgren, H. (ed.), *Proceedings of the 13th International Conference of Computational Linguistics*, Vol. 3. Helsinki 1990, 168-173.

Karlsson , F., Voutilainen, A., Heikkilä , J. & Anttila, A. 1995. *Constraint Grammar: A Language-Independent Framework for Parsing Unrestricted Text.* Mouton de Gruyter, Berlin/New York.

Karttunen, L. & Beesley, K. R. 2005 (this volume). Twenty-five years of finite-state morphology.

Koskenniemi, K. 1983. *Two-level morphology: A general computational model for word-form recognition and production.* Publication 11, University of Helsinki, Department of General Linguistics, Helsinki.

Koskenniemi, K. & Church, K. W. 1988- Two-level Morphology and Finnish. In: *Proceedings of the 12th International Conference on Computational Linguistics*, COLING-88, Budapest, ed. D. VARGHA, 1, pp. 335-339, John von Neumann Society for Computing Sciences, Budapest (1988). B2U.

Koskenniemi, K. & Haapalainen, M.. 1994. GERTWOL: Ein System zur automatischen Wortformerkennung deutscher Wörter. In: Hausser, R. 1996. *Linguistische Verifikation. Dokumentation zur Ersten Morpholympics 1994*, pp. 121-140. Max Niemayer, Tübingen. Also available at URL http://www.lingsoft.fi/doc/gertwol/intro/gertwol.txt.

Kataja, L. & Koskenniemi, K. 1988, Finite-state Description of Semitic Morphology: A Case Study of Ancient Accadian. In: *Proceedings of the 12th International Conference on Computational Linguistics, COLING-88*, Budapest, ed. D. VARGHA, 1, pp. 313-315, John von Neumann Society for Computing Sciences, Budapest (1988). B2U.

Knuuttila, T. (forthcoming in 2006). Harmaalla alueella Monikielisen kieliteknologian yksikkö tutkimuksen ja kaupallistumisen ristipaineessa. In: *Yliopistotutkimuksen muutos ja tietoyhteiskunnan sisäinen ristiriita* [The transformation of academic research and the internal paradox of the information society], Helsinki University Press, Helsinki.

Naumanen, M. 2002. *Nuorten teknologiayritysten menestystekijät* [Success factors of new technology-based companies] (Sitra reports series, ISSN 1457-571X; 28). Edita Publishing Oy, Helsinki. ISBN 951-37-3819-1

Norberg, M. 1988: *Koskenniemen kaksitasomallin mukainen, foneemipohjainen kuvaus ranskan adjektiivien ja substantiivien taivutusmorfologiasta* [A phoneme-based description of the inflectional morphology of French adjectives and nouns according to Koskenniemi's two-level model]. Unpublished Master's thesis, Department of General Linguistics, University of Helsinki.

Ritchie, G., Russell, G., Black, A. & Pulman, S. 1992. *Computational Morphology: Practical Mechanisms for the English Lexicon*. MIT Press, Cambridge, Massachusetts.

Smeaton, A. F., Voutilainen, A. & Sheridan, P. 1990. The application of morpho-syntactic language processing to effective text retrieval. In: *Esprit '90 Conference Proceedings*, pp 619-635, Dordrecht.

Tapanainen P. and Järvinen T. 1997. A non-projective dependency parser. Proceedings of the 5th *Conference on Applied Natural Language Processing* (ANLP'97), pp. 64-71. Association for Computational Linguistics, Washington, D.C.

Vilkki L. 1997. *RUSTWOL: A System for Automatic Recognition of Russian words.* Available at URL: http://www.lingsoft.fi/doc/rustwol/

Vilkki L. 2005 (this volume). RUSTWOL: A Tool for Automatic Russian Word Form Recognition.

2

Inducing a Morphological Transducer from Inflectional Paradigms

LAURI CARLSON

A traditional way to represent the morphology of inflectional languages is through *paradigms*. This paper presents an idea and a program to induce a nondeterministic morphological transducer from traditional style paradigm sets.

2.1 Paradigm morphology

A traditional way to represent the morphology of inflectional languages is through *paradigms*. A paradigm (the Greek word for example) is a list or table of inflectional forms of an example word, representing a given inflectional class. The table is indexed by grammatical tags, and the items in the table cells are inflected forms. In linguistic morphology, this approach is known as the WP (Word and Paradigm) model.

Ideally, to find a given form of a new member of the same class, one substitutes the inflectional stem of the new word in place of the stem of the paradigm word and reads off the resulting form. In grammars intended for human consumption, the relation between the paradigm and its representatives may be subject to simple morphophonological rules sometimes left for the human user to figure out.

Compared to concatenative (IA, Item and Arrangement) morphology, the WP model differs by just listing the forms, without breaking them up into correspondences between individual tags and morphs. For instance, the paradigm of Latin singular nominative noun like *servus* only tells that the genitive plural is *servorum*. Compared to rule (IP, Item and Process) morphology, the corre-

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

spondences do not go down to segment level, so there is no explicit treatment of morphophonology.

Paradigms in a WP morphology can be identified by arbitrary labels (declension or conjugation number) or by a set of thematic forms which suffice to identify the paradigm. For instance, Latin verb *amo* belongs to first conjugation, identified by the thematic form series *amo, amavi, amatum, amare*.

2.2 Inflectional morphology

`murf` is a small Prolog program intended to induce from traditional style paradigm sets a morphological transducer which (1) produces the forms in the paradigms, (2) does not produce any forms either explicitly or implicitly excluded from the paradigms, and (3) generalises common features of the paradigms, reducing redundancy in the paradigms.

The initial idea was quite simple. `murf` reads in forms in a set of tagged forms, trying to place each form in a two-tape finite state network, maximising the match of the new form in the existing network. The new form is matched with the existing network at both ends of the net. A match which leaves the least unmatched residue is chosen, and the missing part is added into the net as a new arc.

Given, for instance, a paradigm

```
Form            Tagging
talossa          talo 1 N SG INE
taloissa         talo 1 N SG INE
talona           talo 1 N SG ESS
```

`murf` correctly infers that the plural essive form is `taloina`:

```
0: talo talo 1 N 4
 4  SG 5
  5 ssa INE 1.
  5 na ESS 1.
 4 i PL 5
5
```

(The number following the base form identifies the base as a member of a given paradigm. The numbering follows that of *Nykysuomen sanakirja* (Dictionary of Contemporary Finnish). As the net shows, `murf` is able to infer a segmentation of the forms into morphs and tags the morphs appropriately. As a side effect of entering the attested form in the network, new, unattested forms may get generated through re-entrances in the net. Call such forms side effects.

The initial idea has gone through a number of refinements to capture famil-

iar morphological phenomena in real data. They include *morphotax, complementary distribution, free variation, blocking, defective paradigms* and *productivity*.

2.2.1 Morphotax

Morphotax concerns the admissible orders of tags in a well-formed word. The heuristics `murf` follows here is that a proposed match of a new word is not allowed to produce unattested taggings.

To guarantee that, `murf` first forms a separate one-tape morphotax network of the taggings it has encountered. When a new form is considered for entry at a given place of the net, its side effects are first checked for morphotax.

2.2.2 Complementary distribution

Complementary distribution is present when any given tagging is realised by just one form, although tags occurring in it have more than one allomorph. For instance, Finnish partitive endings `tA` and `A` are in complementary distribution, the former occurs after heavy syllables and the latter after light ones. Identically tagged forms are *free variants*. `murf` implements complementary distribution by preventing production of free variants as a side effect of insertion.

2.2.3 Free variation

To allow genuine free variation to get past the complementary distribution check, it suffices to tag the variants as different. For instance, Finnish third person possessive suffix has two forms `nsA` and `Vn` which are in free variation after light open syllables. They are tagged as P3/A and P3, respectively. If desired, the distinguishing tags can be merged afterward.

2.2.4 Blocking

Blocking refers to the phenomenon that a lexicalised exception to a regular rule blocks a productive, regular rule. For instance, Finnish nominative plural is `talot`, not `taloi`, as one might be led to expect from the previous data. `murf` accounts for blocking in the following way. When a paradigm is read in, all forms in it are put on a waiting list. Whenever a form is inserted, forms on the waiting list are checked for blocking. An insertion is not allowed if it would produce a side effect blocked by a form on the waiting list.

2.2.5 Defective paradigms

Some paradigms are *defective* in that some forms are missing from an expected cross classification. For instance, Finnish comitative and instructive (instrumental) cases only have one number (plural). From a combinatorial point of view, case and number form in these cases a *portmanteau* morph

instead of two independent morphs. A straightforward way of recording this gap in distribution is to make the tag combination `PL_COM` a tag on its own. Again, the tag can be normalised afterward if needed..

Another distributional gap is that nouns do not occur in comitative plural without possessive suffix (adjectives do). To record such gaps `murf` allows defining, alongside the networks of legitimate forms, separate networks for exceptions. For instance, entry

```
*-      - N PL_COM
```

disallows nouns ending in plural comitative.

2.2.6 Productivity

Productivity refers to the fact that certain forms generalise by default to new words, while others are restricted to a closed set of forms. (This fact is one of the main motivations of paradigm morphology in the first place.) For instance, Finnish nominals have productive vowel stems and less productive consonant stems. A new base form like *pokemon* will automatically go in the productive wovel stem paradigm. `murf` allows marking a variant as a nonproductive one as follows:

```
tienoisiin     tienoo 24 N PL ILL
tienoihin       tienoo 24 N PL_ILL/h!
tienoiden       tienoo 24 N PL GEN
tienoitten      tienoo 24 N PL_GEN/tt!
```

Nonproductive variants marked with `!` will not be generalised into paradigms where they have not been specifically licensed by attested forms.

2.3 Derivational morphology

Derivational morphology allows concatenating base forms coming from different paradigms. A derivational affix may be specific (at least) to part of speech. For instance, Finnish abessive adjective suffix `tOn` produces an adjective out of a noun.

`murf` allows constraining derivational endings with a categorial grammar style tag format `X\Y`

```
onneton onni 8 N tOn N\A 57 A SG NOM
```

This constrains `tOn` to combine with nouns and produce adjectives. (Formally, `X\Y` is analogous to a portmanteau tag discussed above in that it constrains variation at a point in the net.)

2.4 Theory and implementation

Roughly, the idea behind `murf` is to look for a minimal nondeterministic acyclic transducer which produces all of the forms on a list of legitimate

tagged word forms, none of the forms on a list of illicit forms, and produces each legitimate tagging just once. (Free variation is handled by tagging free variants as different and erasing the differences afterward.) The size of the machine is measured by the numbers of states and arcs and the length of the labels on the arcs.

Finding a minimal nondeterministic network for a list of forms is a NP complete problem, Tamm (2004). Furthermore, it is not likely to have a unique solution. There may be more than one minimal machine for any given set of forms, with different side effects.

As stated, the problem is independent of the order of the items on the list(s). `murf`, on the other hand, is order dependent. Its task is to find for each form on the list the smallest extension of the network built so far which produces at least that form, under the same constraints as before. The algorithm is roughly this:

- take a form from list
- while possible
  - find a new minimal arc which generates the form in the network
  - check the arc against constraints on list and net
  - compare the arc to the best find so far
- insert the best arc in network

In `murf`, "smallest extension" is measured in terms of the length of the infix arc. `murf` now actually goes through all minimal infixes which produce the string, testing them against the list and against the network for duplicates, and choosing the shortest among them. This can take long, because there may be many minimal infixes, and each infix may produce many forms which must be compared to the list as well as to the net for duplicates. The minimal infixes depend on the form, so they are computed again for each form.

2.5 Experiments

`murf` has been tested with Finnish nominal and verb paradigms. There are ca 80 nominal and 50 verb paradigms respectively, in the classification of the Dictionary of Contemporary Finnish (original edition). In an initial test, a set of 80 noun paradigms producing around 20,000 forms got coded into a nondeterministic transducer with around 600 states and 1,700 arcs, almost 1,000 of which were epsilon arcs. Only correct forms get produced.

The largest run so far took 34 hours wallclock time. It loaded a training set of about 5,000 word forms to produce a nondeterministic network for about 140 word paradigms, among them all examples of the nominal inflection paradigms listed at front of the Dictionary of Contemporary Finnish, including rare and obsolete patterns. A total of 463,282 word forms get generated from a network of 564 states and 1,647 arcs.

The size of the C runtime `mph` is about 90 kilobytes and runs through the forms in about 4.3 seconds, more than 100 words per millisecond. It stores about 100 words per state, about 5 words per byte.

With little space/time optimisation done in `murf` so far, adding new paradigms gets slow toward the end of the process. There are likely to be ways to make `murf` more time efficient by caching or reordering tasks.

Many forms appear to produce the same illegal side effects. It seems to pay to cache recurrent illegal forms: in a recent test run, the run time went down three fourths as a result.

2.6 Adding new forms to existing paradigms

Adding new words to existing paradigms can be made faster once the paradigm is in. The original idea of `murf` was precisely to implement this insight of the WP model. Thanks to the organisation of the network, new words only need to be given a few thematic forms for the word to settle in the right places in the network, and produce the predictable forms as side effects.

2.7 Guessing forms

It is also possible to use the net to guess the paradigms of unknown words on the basis of thematic forms.

2.8 A runtime morphology analyser/generator in C

The network defined by `murf` can be dumped in the form of a C language string lookup table indexed by a one character lookahead. A tiny (250 lines, 50K) C runtime parser/generator `mph` does lookup from the table at the rate of 100 words per millisecond (raw listing).

2.9 Order sensitivity

murf is sensitive to the order in which forms are presented to it. If regular paradigms are presented before irregular ones, murf tends to overgeneralise, and subregularities across irregular paradigms may get missed. The best strategy seems to be to start with paradigms which exhibit central regularities but make significant splits between regular and irregular sets of endings. In Finnish nouns, a good strategy proved to be to start with bisyllabic nouns in paradigm 40 (`susi 'wolf', vesi 'water'`) whose local cases are regular, while irregular grammatical cases show stem allomorphy.

2.10 Discussion

There are by now a variety of approaches learning morphology from data. Koskenniemi (1991) considers learning two-level morphophonological rules

from surface alternations. Goldsmith (2000) presents a heuristics and a statistical evaluation procedure to find a morphological segmentation for a language from a raw text corpus. Creutz et al. (2005) carry this idea further and make an unsupervised morphological segmenter available for download.

`murf`, in contrast, belongs to the paradigm of supervised learning, as it expects fully tagged and classified paradigms painstakingly prepared and sorted by a linguist, restricting itself to the task of converting the paradigms into a less redundant form. From a linguistic point of view, murf can be seen to implement some of the traditional principles of taxonomical morphemic analysis.

As a solution of the theoretical minimisation problem, `murf` remains naive. More robust methods could be found to optimise the transducer induction task as a purely computational problem. As they are, `murf` and `mph` may just about do for generating small scale morphological analysers and generators for restricted natural language tasks.

References

Creutz, Mathias, Krista Lagus, Krister Linden, and Sami Virpioja. 2005. Morfessor and hutmegs: Unsupervised morpheme segmentation for highly-inflecting and compounding languages. In *Proceedings of the Second Baltic Conference on Human Language Technologies*, pages 107–112. Tallinn.

Goldsmith, Jerry. 2000. *Unsupervised Learning of the Morphology of a Natural Language*. University of Chicago.

Koskenniemi, Kimmo M. 1991. A discovery procedure for two-level phonology. In L. Cignoni and C. Peters, eds., *Computational Lexicology and Lexicography: A Special Issue Dedicated to Bernard Quemada*, pages 451–446.

Tamm, Hellis. 2004. *On Minimality and Size Reduction of One-Tape and Multitape Finite Automata*. Department of Computer Science Series of Publications A,Report A-2004-9. Helsinki: University of Helsinki Department of Computer Science.

3

The DDI Approach to Morphology

KENNETH W. CHURCH

DDI stands for "Don't Do It." White (1980) used the acronym DDI in a delightful paper on garbage collection. He argued 25 years ago that the garbage collector should be run rarely – perhaps once a year – if at all. That's pretty much how many of us work these days. We buy enough memory so we don't have to clean up more than once a week. Rebooting is easier than garbage collecting. As for disks, many of us buy enough to last a couple of years without cleaning up (too much). And when it comes time to clean up, we buy a new machine with even more disk space.

Cleaning up isn't much fun. It would be nice if my son cleaned up his room more often (but then I set a lousy example). Technology isn't the problem or the solution. Even if I came home with a fancy new tool that did most of the work, I bet there would still be many things my son would rather do than clean up his room (if given the choice).

So, what does morphology have to do with garbage collection? White suggests that garbage collecting is harmful. We find that morphological inferences are dangerous. Simple morphological inferences are better than complex inferences. But even simple inferences are worse than none. Like cleaning up, we don't want to do any more morphology than we absolutely have to. Most of us would opt for DDI, if given the choice.

There are lots of programs out there that take out the garbage and decompose words. Some of these programs work remarkably well, for the most part. But, why run such programs, if we don't have to?

There are obvious morphological patterns that we would like our computer programs to take advantage of, but there are also many potential pitfalls. All inferences introduce certain errors, but some inferences are safer than others.

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

To err is human, but to really screw things up, you need a computer.

The argument for morphology comes down to a lack of choice. Do we have to run such programs? It is said that English has rather impoverished morphology compared to other languages such as, say, Finnish. Perhaps so. Thus, we'll use morphology programs for Finnish (because we have to, not because we want to), but for English, we'll use the DDI method, because we can.

However, I wonder if DDI might be an option, even for Finnish. Suppose we cached the more frequent words in the dictionary. How big would the cache have to be? If space is the problem, is morphology the best solution, or are there more effective compression techniques?

3.1 Morphology and the Bell Labs Text-to-Speech (TTS) Synthesizer

I started working on morphology as part of the Bell Labs text-to-speech (TTS) project. I wrote the letter to sound rules. They didn't work very well, maybe 80 percent of the words were ok.

I knew the letter-to-sound rules weren't very good. Rather than fix them, I pushed for the DDI method (dictionary lookup). It wasn't that hard to raise the funds to buy the rights to a dictionary, but there was serious pushback on the memory. It was hard, in those days, to justify an extra 1/4 megabyte of memory.

Dictionary lookup had obvious advantages in terms of precision, compared to letter-to-sound rules. (Fewer inferences/guesses are better than more inferences/guesses.) However, to improve recall, the dictionary had to be extended with (morphological) inferences that are safer than letter to sound rules, but not as safe as DDI (dictionary lookup). In Coker et al. (1990), we evaluated a number of inference methods:

- Stress-Neutral Suffixes (including regular inflection): abandons = abandon + s, abandoning = abandon + ing, abandonment = abandon + ment, Abbotts = Abbott + s, Abelson = Abel + son.
- Primary-Stress Ending: addressee = address + ee, accountability = account + ability, adaptation = adapt + ation.
- *ity*-Class Ending: abnormality = abnormal + ity, Adomovich = Adam + ovich, Abmbrosian = Ambrose + ian.
- *al*-Class Ending: accidental = accident + al, combative = combat + ive.
- Suffix Exchange: nominee = nominate − ate + ee, Agnano = Agnelli − elli + ano, Bierstade = Bierbaum − baum + stadt.
- Prefix: adjoin = ad + join, cardiovascular = cardio + vascular, O'brien = O' + brien, Macdonald = Mac + donald.

- Compound: airfield = air + field, anchorwoman = anchor + woman, Abdulhussein = Abdul + hussein, Baumgaertner = Baum + gaertner.
- Rhyming: Plotsky (from Trotsky), Alifano (from Califano).

Tables 1 and 2 report coverage by token over two data sources:

- 1988 Associated Press (AP) newswire, plus
- a list of names from Donnelley Marketing.

Names are distinguished from non-names (general vocabulary) because names are relatively hard. A high-quality commercial dictionary was used for general vocabulary. In addition, there was a special purpose dictionary of 50,000 common American surnames.

Table 2 reports precision by morphological method. A single judge listened to approximately 100 names in each row and labeled them as:

- Good: That's how I would have said it.
- OK: I could imagine someone else saying it that way or I don't know.
- Poor: I know that's wrong.

Don't make an error-prone inference if you don't have to. Even DDI (dictionary lookup) is far from perfect. The judge labeled 2 percent of names in the surname dictionary as "poor."

If you have to make a morphological inference, focus on simple, safe inferences with high precision and recall. The stress-neutral case, which includes regular inflection, is simpler than many, but even so, the judge labeled 4 percent of them as "poor," twice as many as DDI.

In addition to morphological inferences, we also evaluated the rhyming inference, proposed by Byrd and Chodorow (1985). Rhyming is riskier than DDI and conservative morphological processes (such as regular inflection), but safer than aggressive morphology such as compounding. Compounding is nearly as risky as letter to sound rules (only 83 percent good). And that is about as bad as it gets.

3.2 Information Retrieval

Information Retrieval was one of the first fields to question the value of morphological inferences. Do stemmers help retrieval performance? See Table 8.1 in Frakes and Baeza-Yates (1992) for a summary of stemming experiments, many of which failed to find much of a difference in terms of precision and recall. Salton and Lesk (1968) conclude, "For none of the collections is the improvement of one method over the other really dramatic, so that in practice either procedure might reasonably be used."

Such a mixed bag of mostly negative results ought to be disturbing for those of us working in natural language processing. If it is hard to show that something as simple as stemming is helpful, how can we possibly justify our

TABLE 1 Simple inferences cover much of the Associated Press news.

Inference Method	Ordinary Words (Non-names)	Capitalized Words (Names)
Direct Hit (DDI)	75%	70%
Stress Neutral	17%	14%
ity-class	1%	2%
al-class	1%	1%
Rhyme	0%	2%
Prefix	2%	0%
Compound	1%	1%
Combinations	4%	8%
All Dictionary-Based Methods	100%	97%
Letter to Sound Rules	0%	3%

TABLE 2 Simple inferences are relatively reliable (on names in Donnelly Marketing List). Rows are sorted by the "Poor" column.

Method	Good	OK	Poor	Coverage
Direct Hit (DDI)	95%	3%	2%	60%
Suffix Exchange	93%	5%	3%	1%
Stress Neutral	91%	6%	4%	25%
Rhyme	88%	8%	4%	2%
ity-Class	91%	3%	6%	1%
al-Class	87%	8%	6%	1%
Compound	83%	3%	14%	2%
All Dictionary-Based Methods				98%
Letter to Sound Rules				2%

interests in more challenging forms of natural language processing such as part of speech tagging, word sense disambiguation and parsing?

In Church (1995), I argued that morphological variants like "hostage" and "hostages" should not be treated as one term, or two, but somewhere in between, perhaps a term and a half. I looked at the distribution of terms across documents in the AP newswire, as illustrated in Table 3. In the 1988 Associated Press Newswire, there were 619 documents that mentioned both "hostage" and "hostages," 479 documents that mentioned the former but not the latter, 648 that mentioned the latter but not the former, and 78,223 that mentioned neither. One can measure similarity statistics based on such contingency tables. The correlation of documents that mentioned "hostage" and documents that mention "hostages" is about $\frac{1}{2}$, well above 0, but well below 1.

Treating "hostages" and "hostage" as one term makes sense, under the

TABLE 3 Contingency Table (1988 Associated Press Newswire)

	hostages	¬hostages
hostage	619	479
¬hostage	648	78,223

vector space model, if the correlation is 1. Treating them as two terms would be appropriate if the correlation were 0. But the correlation is in between. The mixed bag of results arises, I suspect, because neither the one term simplification, nor the two term simplification, fits the data. In the spirit of "do no harm," I lean toward DDI (two terms), rather than conflate things that shouldn't be conflated.

There are some very interesting lexical patterns to these correlations. Some words have large correlations with their variants, and some don't. Nouns and words with "lots of content" (better keywords for information retrieval) tend to have higher correlations with their variant forms than function words and non-nouns with relatively little meaning.

- large correlations: hostage(s), reactor(s), rebel(s), guerrilla(s), abortion(s), delegate(s), drug(s), stock(s), pesticides(s), airline(s).
- small correlations: await(s), ground(s), possession(s), boast(s), belonging(s), compare(s), direct(s), shield(s), last(s), urge(s).

The correlations are remarkably stable over data collections. If a pair has a large correlation in one corpus, then the correlation tends to be large in other corpora. In Church (1995), I studied correlations of correlations across five years of the AP newswire for 999 pairs of words like "hostage" and "hostages" that differ by a final "s." If we know the correlation of these 999 pairs in one year of the AP, then we can account for 80 percent or more of the variance in predicting the correlations of these 999 pairs in another year.

Morphology is not that different from case normalization and other text normalization steps that are commonly used in practice, but may do more harm than good. Some pairs like *Hurricane*/*hurricane* have large correlations while others like *Continental*/*continental* and *Anytime*/*anytime* do not. The correlations are large when both members of the pair have a lot of meaning and they refer to the same thing (*Hurricane*/*hurricane*); the correlations are small when they refer to different things (*Continental*/*continental*)[1] or not much of anything (*Anytime*/*anytime*).

- Large correlations (between upper and lower case variants): Hurricane, Pope, Emperor, Lottery, Zoo, Ballet, Golf, Canal, Immigration.

[1] *Continental* is an airline; *continental* is not.

- Small correlations (between upper and lower case variants): Troy, Path, Editions, Continental, Burns, Levy, Haven, Rush, Anytime

Morphology and case normalization are similar to priming and repetition. If a document mentions a word once, then it is likely that we will see that word (and its friends) again in the same document, especially if the word is a good keyword with lots of meaning. In Church (2000), I looked at the distribution of "Noriega" across documents. Noriega was mentioned in quite a number of AP news articles (about 6 articles per thousand) when the US invaded Panama. Under standard independence assumptions, the chance of two Noriega's should be $(6/1000)^2$. We shouldn't expect to find an article with two or more Noriega's, but we have lots of them. Of those documents that mention Noriega at least once, 75 percent mention Noriega a second time.

Repetition of a word (and its morphological variants and other friends) is more likely for good keywords with lots of content and less likely for function words that aren't very useful for information retrieval. In Church (2000), I showed that distinctive surnames are more likely to be repeated than ordinary first names:

- Distinctive Surnames: Noriega, Aristide, Escobar
- Ordinary First Names: John, George, Paul

There are lots of important linguistic generalizations that produce undeniable distributional patterns. Morphology is one of many such factors. The information retrieval community had hoped that they could use standard off-the-shelf morphology programs in straightforward ways to take advantage of morphological patterns to improve precision and recall. Such attempts have not worked out very well.

In the spirit of "there is no data like more data," it is probably wise not to collapse morphological variants, unless you run out of data, and have no choice. If we have enough data to compute the distribution of each form of each lemma separately, we might as well do so. As long as we have plenty of data, there is no need to introduce unnecessary assumptions like perfect correlation or total independence.

In recent years, with all the excitement about the web, there has been more talk about what we can do with all the data we have, and less talk about running out of data. Of course, we never have enough data, but right now, there is more interest in finding clever ways to take advantage of all the data we have, and less interest in finding clever smoothing techniques to compensate for all the data we don't have. Right now, the glass is half full, not half empty.

3.3 Part of Speech Tagging

There are, of course, many other applications for morphology programs including part of speech tagging and spelling correction. Statistical part of

speech taggers make use of two sets of probabilities:

- Lexicon: $Pr(tag|word)$
- Context: $Pr(tag|context)$

Much of the literature has tended to concentrate on the context model, but that's the easy part. Typically there are only P^3 parameters in the context model, where P is the number of parts of speech, typically several dozen. The lexicon is far more challenging, since there are more parameters: $V \times P$, where V is the size of the vocabulary, typically between 10^5 and 10^6. V is typically much larger than P^2.

To illustrate the challenge with lexical probabilities, my favorite example is the word "yawn." "Yawn" occurs once in the training corpus (the Brown Corpus)[2] as a noun, and once as a verb. Based on this sparse evidence, we need to know, not only the probability that "yawn" could be a noun or a verb, but also the probability that it could be an adjective, or any other part of speech. Just because we haven't seen "yawn" as an adjective, doesn't mean it can't happen.

In fact, most words are like "yawn." Of the 50,000 words in the Brown Corpus, 80 percent appear 5 times or less. Given that we have more than 5 parts of speech, we have more parameters than data for most words. Perhaps the Brown Corpus is just too small, but if we collect a larger corpus, we'll discover a pile of new infrequent words. The more data we look at, the more we realize just how little we know.

One might hope that one could infer the lexical probabilities by reasoning across morphological variants, but I have never been able to make this work. In Church (1992), section 5.2, I looked at noun/verb ambiguous words like "yawn" and "yawns." Some of them are more likely to be nouns and some are more likely to be verbs. I was hoping that the probabilities of one morphological variant could be used to infer the probabilities for the other morphological variant, but I failed to find anything of use. While there are clear constraints on the grammatical possibilities, the statistical probabilities are less predictable.

Co-training (Blum and Mitchell, 1998), appears to be a more promising direction forward. That is, there are contexts where the tagger is likely to do relatively well. For example, after "the," we are much more likely to see a noun than a verb. It ought to be possible to run the tagger on vast quantities of untagged material and use observations based on this untagged data to estimate parameters that we couldn't estimate very well based on the relatively small amount of labeled training data that we happened to have, the one-million word tagged Brown Corpus.

[2] http://en.wikipedia.org/wiki/Brown_Corpus

Co-training needs to be performed carefully. Merialdo (1994) reported that performance degraded the more he iterated (unless he started with almost no labeled training data). Iterating always increases the model's likelihood scores, but doesn't always improve performance. Co-training should not be allowed to move the parameter settings very far from the estimates based on the labeled training data, especially when we have plenty of labeled training data for the parameter in question. With appropriate precautions, I believe, co-training will be more effective than morphological inference for estimating lexical probabilities.

3.4 Spelling Correction

These days, it is no longer as necessary as it used to be to use morphology and other tricks to reduce the size of the lexicon. McIlroy (1982) describes the heroics that were used in the original Unix Spell program to pack a small dictionary of 32,000 English words into a PDP-11 address space of 64k bytes. That's just two bytes per word.

Normally, hash tables use a hash function to jump quickly to the appropriate bucket, and then the key is checked more thoroughly against the keys in the bucket. But the Unix Spell program didn't have enough memory to store the keys in the table, so they didn't. It was necessary to introduce a lossy compression method that worked remarkably well in practice. Suppose we have a table of $N \approx 32,000$ words. Choose a prime, P, that is somewhat larger than N. P trades off compression for accuracy. Shrinking P saves memory, but introduces loss (false hits). Increasing P reduces loss, but consumes memory.

- Memory: $N[\frac{1}{log_e 2} + log_2 \frac{P}{N}]$ bits to store N words
- Loss: Pr(false hit) = $1 - (1 - 1/P)^N$

The Unix Spell program hashed each word in the dictionary and then mod the hash code by P, to obtain a number between 0 and $P-1$. If the hash function is chosen well, these numbers will have a Poisson distribution. Sort the hash codes and take first differences. These differences will be exponentially distributed with a mean of P/N, since the inter-arrivals of a Poisson process are exponential. The differences were then compressed using a Golomb code, an optimal Huffman code for exponentially distributed values.

During runtime, an input word would be hashed into a hash code from 0 to $P - 1$, using the procedure described above. This hash code would then be looked up in the compressed table (by performing the Golomb decode and computing running sums to invert the first differences). If the hash code was found in the table, then the program assumed the input word was correctly spelled. If not, the input word was flagged as potentially misspelled. There is a small probability, Pr(false hit), that an incorrectly spelled word would generate a false hit, hashing into the same place as some other correctly spelled

word, causing the Unix Spell program to fail to flag a misspelled word that it would have flagged, if not for the clever (but lossy) compression.

In addition to these heroic compression techniques, the Unix Spell program made heavy use of whatever tricks it could, including morphology; memory was unbelievably tight. According to McIlroy (1982), derived words were culled from the dictionary. Thus, "Wells" was culled because it could be analyzed as "Well" + "s." "Peters" was analyzed as "Peter" + "s." The culling was recursive, so "Peter" was also culled since it could be decomposed into "Pete" + "er"! Recursive application of such heuristics was risky but necessary, given the lack of memory.

These heroic compression methods are no longer necessary now that we have 32-64 bits of address space. Modern spelling correctors no longer need to make use of fancy (lossy) compression techniques. They no longer cull the dictionary as aggressively. Modern spelling correctors have larger dictionaries with 10^5 - 10^6 entries, many of which could be derived from other entries, but doing so would introduce (unnecessary) risk. Sometimes it is ok to add "er" to some other lexical entry, and sometimes it isn't.

3.5 Conclusion

There are lots of morphology programs out there, many of which work surprisingly well. Nevertheless, for many practical applications, we prefer not to use such programs, if we have the choice. Simple morphological inferences are better than complex inferences. But even simple inferences are worse than none.

References

Blum, Avrim and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the Eleventh Annual Conference on Computational Learning Theory (COLT)*, pages 92–100. New York: ACM Press.

Byrd, Roy J. and Martin S. Chodorow. 1985. Using an on-line dictionary to find rhyming words and pronunciations for unknown words. In *Proceedings of the 23rd conference on Association for Computational Linguistics (ACL)*, pages 277–283.

Church, Kenneth. 1992. Current practice in part of speech tagging and suggestions for the future. In A. Mackie, T. McAuley, and C. Simmons, eds., *In Honor of Henry Kucera*, pages 13–48. University of Michigan: Michigan Slavic Studies.

Church, Kenneth. 1995. One term or two? In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 310–318. New York: ACM Press.

Church, Kenneth W. 2000. Empirical estimates of adaptation: The chance of two noriegas is closer to $p/2$ than p^2. In *Proceedings of the 17th conference on Computational linguistics (COLING)*, pages 180–186.

Coker, Cecil, Kenneth Church, and Mark Liberman. 1990. Morphology and rhyming: two powerful alternatives to letter-to-sound rules for speech synthesis. In *ESCA Workshop on Speech Synthesis*, pages 83–86.

Frakes, William B. and Ricardo Baeza-Yates, eds. 1992. *Information retrieval: data structures and algorithms*. Upper Saddle River, NJ: Prentice-Hall, Inc. ISBN 0-13-463837-9.

McIlroy, M. D. 1982. Development of a spelling list. *IEEE Transactions on Communications* 30:91–99.

Merialdo, Bernard. 1994. Tagging english text with a probabilistic model. *Computational Linguistics* 20(2):155–171.

Salton, G. and M. E. Lesk. 1968. Computer evaluation of indexing and text processing. *J. ACM* 15(1):8–36.

White, Jon L. 1980. Address/memory management for a gigantic lisp environment or, gc considered harmful. In *LFP '80: Proceedings of the 1980 ACM Conference on LISP and Functional Programming*, pages 119–127. New York: ACM Press.

4

Finite-State Parsing of German

ERHARD W. HINRICHS

4.1 Introduction

There has been a remarkable revival of finite-state methods in linguistics over the last twenty-five years. This renewed interest is a direct consequence of the pioneering work on two-level phonology and morphology by Kimmo Koskenniemi (Koskenniemi, 1983) and of the independently developed approach to finite-state morphology by Ron Kaplan and Martin Kay (Kaplan and Kay, 1994). Based on mathematically rigorous models of finite-state transduction, there are now wide-coverage finite-state accounts of an impressive range of typologically diverse languages available. Inspired by these successes, research of finite-state models for syntactic analysis was revived in the early nineties, notably by Stephen Abney (Abney, 1991) and by Fred Karlsson and his associates (Karlsson et al., 1995)[1]. Their research ended a period of more than three decades of little or no research on finite-state models of syntax under the influence of Chomsky's claim that finite-state automata are inadequate due to their inability to account for center-embedding construction in natural languages (Chomsky, 1963).

The two alternative models of finite-state syntax developed by Abney and Karlsson reflect in an interesting way two leading paradigms for representing syntactic structure. Abney's chunk parser is designed to provide a partial bracketing of an input text. This bracketing identifies non-recursive phrases, so-called *chunks*, which span from the left periphery of a phrase to its phrasal head. The resulting bracketing is partial in that it leaves any structural rela-

[1] Rules in Constraint Grammar are, in isolation, implementable with finite-state methods. Editor's comment.

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

tionships between individuals chunks unresolved.

Karlsson's constraint-grammar formalism is designed to provide a shallow syntactic parse of an input text that identifies the beginnings and ends of non-recursive phrases and the grammatical functions of verbal complements.

The purpose of this paper is to review recent work on finite-state syntactic analysis of German. Rather than comparing the details of individual finite-state parsing systems for German, the discussion will focus on those aspects of German sentence structure that make German an interesting language from a finite-state perspective. Section 4.2 surveys existing finite-state and constraint-based parsers of German. Section 4.3 discusses complex prenominal modifier structures in German which are recursive in nature. Their recursiveness provides an interesting challenge for Abney's conception of what a chunk is. Section 4.4 gives an overview of the main characteristics of German sentence structure This provides the necessary background for the discussion of interesting challenges and opportunities that the sentence structure of German poses for finite-state approaches. This discussion is the topic of section 4.5.

4.2 A Survey of Finite-State and Constraint-Grammar Parsers of German

Most of the research on finite-state parsing of German has utilized Abney's chunk parsing model and produces partial bracketings of the input text. Two recent examples of Abney-style chunk parsers for German are the Dereko parser (Müller and Ule, 2001) and the YAP parser (Kermes, 2002, Kermes and Evert, 2002). In addition to finite-state chunk parsers, Schmid and Schulte im Walde (2000) have developed a statistical chunk parser for German that is based on probabilistic context-free grammars.

There are at least four parsers for German that use finite-state methods internally and produce dependency relations. Connexor, a Finnish language technology company, has developed a syntactic parser called *Machinese Syntax* for a variety of languages, including German, that produce part-of-speech classes, inflectional tags, noun phrase markers and syntactic dependencies for written input. The output representations follow the style of annotation familiar from Constraint Grammar.[2] Schiehlen (2003) has developed a finite-state parser for German that produces dependency relations and that uses underspecification to encode ambiguities that arise from alternative valence frames of verbs and from alternative attachment sites for PP modifiers. Most recently, Trushkina (2004) and Müller (2005) have developed parsers that combine chunk parsing with dependency parsing. Trushkina's GRIP parser is based

[2] Duchier (1999) and Foth et al. (2004) have also developed dependency parsers for German, albeit without explicitly relying on finite-state methods.

on the Xerox Incremental Parsing System (XIP) (Aït-Mokhtar et al., 2002), while Müller's parser uses the suite of lcg tools (Mikheev et al., 1998, 1999). Both parsers are limited to those dependency relations that refer to complements and do not deal with adjuncts.

4.3 Prenominal Modifiers and Recursive Chunk Structures

One of the syntactic constructions that make German an interesting language from a chunk parsing perspective are complex prenominal modifiers such as the participial construction as in (1a).

(1) a. der seinen Sohn liebende Vater
the his son loving father
'the father who loves his son'

b. $[_{NC}$ der $[_{NC}$ seinen Sohn] liebende Vater]

Such examples are interesting since they do not simultaneously satisfy the two defining properties that Abney associates with the term *chunk*. Abney (1996) defines the notion of chunk as " ... the non-recursive core of an intra-clausal constituent, extending from the beginning of the constituent to its head." In the case of (1a), the article *der* and the nominal head *Vater* seem to represent the left and right periphery of a nominal chunk. However, chunks are also defined as non-recursive structures. This seems to suggest that only the substring *seinen Sohn* qualifies as a noun chunk (NC) and seems rule out the structure in (1b), where the entire string is a nominal chunk as well. In fact, Abney appeals to the "no chunk within a chunk"-constraint to explain the ungrammaticality of English NPs as in (2).

(2) * the proud of his son father

For cases like (1), there seem to be two solutions to this impasse: one may argue that only the NP inside the premodifier, or one considers the complex NP as a nominal chunk and gives up.

A telling piece of evidence in favor of the latter solution is provided by the grammaticality of (3), the German counterpart of (2).

(3) der auf seinen Sohn stolze Vater
the on his son proud father
'the father who is proud of his son'

This seems to suggest that Abney's "no chunk within a chunk"-constraint is not universally applicable across languages, even though it does seem to hold for English. However, the assumption that chunks are non-recursive in nature is not only motivated by examples such as (2). Notice that once one accepts recursive bracketings shown in (1), one allows center-embedding constructions. The fact that natural languages allow for such constructions was

identified by Chomsky as the key argument for rejecting finite-state models for natural language analysis. If one allows center-embeddings to an arbitrary level of embedding, then the analysis of such constructions lies beyond the expressive power of regular grammars. Chomsky's argument crucially rests on the assumption that there is in principle no depth bound on the number of embeddings inside a center-embedding construction. Chomsky readily admits that there are, of course, processing limitations by language users that limit center-embeddings to two or at most three for a given utterance. However, such upper bounds, he argues, should be considered aspects of performance grammar, not of competence grammar. If one accepts this argument then the inadequacy of finite-state grammars seems to refer to competence grammar only. Thus, if one views a finite-state parser as a model of performance grammar, then one can simply impose a reasonable depth bound on center-embedding constructions in a finite-state grammar. This is precisely what Kermes (2002) and Müller (2005) have done in order to be able to treat complex prenominal modifiers as part of chunks that exhibit limited, i.e. depth-bounded, recursion. In addition to complex, prenominal modifiers, Kermes' YAC parser also admits a limited number of post-head nominal modifiers as in (4).

(4) a. die Köpfe der Apostel
the heads of the apostles
'the heads of the apostles'

b. Jahre später
years later
'year later'

In order to accommodate examples such as (1), (3), and (4), Kermes (2002) modifies Abney's definition of a chunk as in (5).

(5) A chunk is a continuous part of an intra-clausal constituent including recursion, pre-head as well as post-head modifiers, but no PP-attachment or sentential elements.

4.4 The Macro-structure of German: topological fields

One of the characteristic features of German syntax is the placement of the finite verb in different clause types. Consider the finite verb *wird* in (6) as an example.

(6) a. Peter wird das Buch gelesen haben.
Peter will the book read have
'Peter will have read the book.'

b. Wird Peter das Buch gelesen haben?
Will Peter the book have read
'Will Peter have read the book?'

c. dass Peter das Buch gelesen haben wird.
that Peter the book read have will
'... that Peter will have read the book.'

In non-embedded assertion clauses, the finite verb occupies the second position in the clause, as in (6a). In yes/no questions, as in (6b), the finite verb appears clause-initially, whereas in embedded clauses it appears clause finally, as in (6c). Regardless of the particular clause type, any cluster of non-finite verbs, such as *gelesen haben* in (6a) and (6b) or *gelesen haben wird* in (6c), appears at the right periphery of the clause.

The discontinuous positioning of the verbal elements in verb-first and verb-second clauses is the traditional reason for structuring German clauses into so-called *topological fields* (Erdmann, 1886, Drach, 1937, Höhle, 1986). The positions of the verbal elements form the *Satzklammer* (sentence bracket) which divides the sentence into a *Vorfeld* (initial field), a *Mittelfeld* (middle field), and a *Nachfeld* (final field). The Vorfeld and the Mittelfeld are divided by the *linke Satzklammer* (left sentence bracket), which is realized by the finite verb or (in verb-final clauses) by a complementizer field. The *rechte Satzklammer* (right sentence bracket) is realized by the verb complex and consists of verbal particles or sequences of verbs. This right sentence bracket is positioned between the Mittelfeld and the Nachfeld. Thus, the theory of topological fields states the fundamental regularities of German word order.

The topological field structures in (7) for the examples in (6) illustrate the assignment of topological fields for different clause types.

(7) a. $[_{VF}$ $[_{NC}$ Peter]] $[_{LK}$ wird] $[_{MF}$ $[_{NC}$ das Buch]]
$[_{RK}$ $[_{VC}$ gelesen haben.]]

b. $[_{LK}$ Wird] $[_{MF}$ $[_{NC}$ Peter] $[_{NC}$ das Buch]]
$[_{RK}$ $[_{VC}$ gelesen haben?]]

c. $[_{LK}$ $[_{CF}$ dass]] $[_{MF}$ $[_{NC}$ Peter] $[_{NC}$ das Buch]]
$[_{RK}$ $[_{VC}$ gelesen haben wird.]]

(7a) and (7b) are made up of the following fields: LK (linke Satzklammer) is occupied by the finite verb. MF (Mittelfeld) contains adjuncts and complements of the main verb. RK (rechte Satzklammer) is realized by the verbal complex (VC). Additionally, (7a) realizes the topological field VF (Vorfeld), which contains the sentence-initial constituent. The left sentence bracket (LK) in (7c) is realized by a complementizer field (CF) and the right sentence bracket (RK) by a verbal complex (VC) that contains the finite verb *wird*.

4.5 Finite-state Parsing of German

The structure of topological fields delineates the borders and the composition of a clause and thus reveals the overall anatomy of a sentence. It turns out

that topological fields together with chunked phrases provide a solid basis for a robust analysis of German sentence structure. All chunk parsing systems mentioned in section 4.2 adopt an annotation strategy which annotates the topological fields for the left and right sentence brackets before identifying any other fields or chunks. To my best knowledge, this strategy was first proposed by Braun (1999) and by Neumann et al. (2000) as a means of identifying sentence boundaries for German.

It turns out that the advantages of topological field annotation go significantly beyond sentence boundary detection. Robust identification of topological fields can help reduce the search space for subsequent chunk annotation since chunks can only occur within the boundaries of a given topological field.

(8) [$_{VF}$ [$_{NC}$ Außenminister Joschka Fischer]] [$_{LK}$ hat] [$_{MF}$ [$_{NC}$ die Abgeordneten]] [$_{RK}$ gebeten] [$_{NF}$ [$_{MF}$ [$_{NC}$ die Entscheidung]] [$_{RK}$ zu verschieben.]]

'Foreign Minister Joschka Fischer asked the members of parliament to postpone the decision.'

(8) is a V2-clause with an extraposed *zu*-infinitive that is governed by the verb form *gebeten*. Such extraposed constituents are positioned in the topological field Nachfeld (NF). By locating the noun chunks *die Abgeordneten* and *die Entscheidung*, which occurs the Mittelfeld of the V2-clause, in different topological fields, it becomes clear that they modify the verbs *gebeten* and *verschieben*, respectively.

Recognition of topological fields can also effectively reduce potential ambiguities that can arise if only local syntactic context is taken into account.

(9) [$_{VF}$ [$_{NC}$ Man]] [$_{LK}$ sah] [$_{MF}$ [$_{PC}$ in [$_{NC}$ der Öffentlichkeit]] [$_{ADVC}$ nur] [$_{NC}$ Männer] [$_{PC}$ mit [$_{NC}$ Zigarette]]] [$_{KOORD\_F}$ und] [$_{VF}$ [$_{NC}$ rauchende Frauen]] [$_{LK}$ waren] [$_{MF}$ [$_{NC}$ ein Thema] [$_{PC}$ für Karikaturen]]

'In public, you saw only men with cigarettes, and smoking women were a topic for caricatures.'

In (9) the coordination *und* forms a coordination field (KOORD_F) with two V2 clauses as sentential conjunctions. The parallelism between the two clauses can be easily detected in terms of the their left and right sentence brackets and their Vorfeld constituents. However, if only local context is taken

into account, the coordination may be misanalysed as an NP conjunction between the two NP chunks *Männer mit Zigarette* and *rauchende Frauen*.

Identifying the left and right sentence bracket of a clause prior to any other syntactic chunk annotation follows the principle of "easy first" parsing advocated by Abney since these two sentence brackets can be detected with great reliability for any clause type of German.

As shown by Müller and Ule (2001), Hinrichs et al. (2002), Müller and Ule (2002), another class of ambiguities that can be resolved by topological field information concerns potential ambiguities in part-of-speech assignments to lexical tokens. Two classes of common tagging errors in German concern the distinction between finite and non-finite verb forms and the distinction between homonymous prepositions and subordinating conjunctions.[3] The token *seit* in (10) is ambiguous between a preposition (APPR) or a subordinating conjunction (KOUS).[4]

(10) $[_{VF}$ $[_{LK}$ $[_{CF}$ Seit]] $[_{MF}$ Banting und Best Insulin zum ersten Mal] $[_{RK}$ $[_{VC}$ isolieren konnten]]], $[_{LK}$ haben] $[_{MF}$ die Mediziner lebenserhaltende Kontrolle über Diabetiker] $[_{RK}$ $[_{VC}$ gewinnen können]].

'Ever since Banting and Best have been able to isolate insulin for the first time, physicians have been able to win life-preserving control of diabetes.'

The theory of topological fields helps to determine the correct tag for *seit* in such cases. The entire clause is a verb-second clause with an embedded clause occupying the clause-initial position. The embedded clause has to adhere to the constraints on how the left and right sentence bracket have to be realized for a verb-final clause. In particular, the left sentence bracket (LK) has to consist of a complementizer field (CF) which can be realized by a coordinating conjunction (KOUS), but crucially not by a preposition (APPR).

Sentence (11) provides an example of a potential part-of-speech ambiguity between a finite (VVFIN) and a non-finite (VVINF) verb for the verb form *nehmen*.

(11) $[_{VF}$ $[_{NC}$ Libyen]] $[_{LK}$ kann] $[_{MF}$ $[_{NC}$ keinen Einfluss] $[_{PC}$ auf $[_{NC}$ die Politik]] $[_{NC}$ Marokkos]] $[_{RK}$ nehmen]

'Libya can exert no influence on the politics of Marocco.'

[3] See Brants (1999) for more detailed discussion.

[4] The part-of-speech tags used for the annotation are taken from the Stuttgart-Tübingen tagset (STTS) Schiller et al. (1995).

Once again, the topological field assignment shown in (11) uniquely determines that *nehmen* has to be a non-finite verb (VVINF) since the right sentence bracket in a verb-second clause may only contain non-finite verbs.

Notice that the type of topological field information that resolves the two types of part-of-speech ambiguities illustrated by examples (10) and (11) are non-local in nature. The crucial clues for disambiguating the lexical tokens in question span essentially the entire clause. It is for this very reason that such examples pose a serious challenge for both rule-based and statistical taggers.[5]

4.6 Conclusion

This paper has presented a survey of finite-state parsing systems for German and has discussed two aspects of German sentence structure that are of general interest from a finite-state perspective: the treatment of complex prenominal modifiers and the characterization of German clauses structure in terms of topological fields.

References

Abney, Steven. 1991. Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny, eds., *Principle-Based Parsing*. Dordrecht: Kluwer Academic Publisher.

Abney, Steven. 1996. Chunk stylebook. Unpublished manuscript, University of Tübingen.

Aït-Mokhtar, Salah, Jean-Pierre Chanod, and Claude Roux. 2002. Robustness beyond shallowness: incremental deep parsing. *Natural Language Engineering* 8(2–3):121–144.

Brants, Thorsten. 1999. *Tagging and Parsing with Cascaded Markov Models*, vol. 6 of *Saarbrücken Dissertations in Computational Linguistics and Language Technology*. DFKI, Universität des Saarlandes.

Braun, Christian. 1999. *Flaches und robustes Parsen deutscher Satzgefüge*. Diplomarbeit, Universität. des Saarlandes, Saarbrücken.

Chomsky, Noam. 1963. Formal properties of grammars. In R. D. Luce, R. R. Bush, and E. Galanter, eds., *Handbook of Mathematical Psychology*, vol. II, pages 323–418. John Wiley.

Drach, Erich. 1937. *Grundgedanken der Deutschen Satzlehre*. Frankfurt/M.: Diesterweg.

Duchier, Denys. 1999. Axiomatizing Dependency Parsing Using Set Constraints. In *Proceedings of the Sixth Meeting on Mathematics of Language (MOL 6)*, pages 115–126. Orlando, FL.

Erdmann, Oskar. 1886. *Grundzüge der deutschen Syntax nach ihrer geschichtlichen Entwicklung dargestellt*. Stuttgart: Verlag der Cotta'schen Buchhandlung. Erste Abteilung.

[5] For a more detailed discussion see Trushkina and Hinrichs (2004).

Foth, Kilian, Michael Daum, and Wolfgang Menzel. 2004. A broad-coverage parser for German based on defeasible constraint. In *KONVENS 2004, Beiträge zur 7. Konferenz zur Verarbeitung natürlicher Sprache*, pages 45–52. Vienna.

Hinrichs, Erhard W., Sandra Kübler, Frank H. Müller, and Tylman Ule. 2002. A hybrid architecture for robust parsing of German. In *Proceedings of LREC 2002*. Las Palmas, Gran Canaria.

Höhle, Tilman. 1986. Der Begriff "Mittelfeld", Anmerkungen über die Theorie der topologischen Felder. In *Akten des Siebten Internationalen Germanistenkongresses 1985*, pages 329–340. Göttingen, Germany.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20(3):331–378.

Karlsson, Fred, Atro Voutilainen, Juha Heikkilä, and Arto Anttila. 1995. *Constraint Grammar, A Language-independent System for Parsing Unrestricted Text*. Mouton de Gruyter.

Kermes, Hannah. 2002. *Off-line (and On-line) Text Analysis for Computational Lexicography*. Ph.D. thesis, University of Stuttgart, Institut für Maschinelle Sprachverarbeitung.

Kermes, Hannah and Stefan Evert. 2002. YAC – a recursive chunker for unrestricted German text. In M. G. Rodriguez and C. P. Araujo, eds., *Proceedings of the Third International Conference on Language Resources and Evaluation*, vol. V, pages 1805–1812.

Koskenniemi, Kimmo. 1983. Two-level model for morphological analysis. In *IJCAI-83*, pages 683–685. Karlsruhe, Germany.

Mikheev, Andrei, Claire Grover, and Marc Moens. 1998. Description of the LTG system used for MUC-7. In *Proceedings of 7th Message Understanding Conference (MUC-7)*.

Mikheev, Andrei, Claire Grover, and Marc Moens. 1999. XML tools and architecture for named entity recognition. *Markup Languages: Theory and Practice* 1(3):89–113.

Müller, Frank Hendrik. 2005. *A Finite-State Approach to Shallow Parsing and Grammatical Functions Annotation of German*. Ph.D. thesis, University of Tübingen.

Müller, Frank Henrik and Tylman Ule. 2001. Satzklammer annotieren und Tags korrigieren. Ein mehrstufiges Top-Down-Bottom-Up-System zur flachen, robusten Annotierung von Sätzen im Deutschen. In H. Lobin, ed., *Proceedings der GLDV-Frühjahrstagung 2001*, pages 225–234. Gießen. Sig.:sb.

Müller, Frank H. and Tylman Ule. 2002. Annotating topological fields and chunks – and revising pos tags at the same time. In *Proceedings of the Nineteenth International Conference on Computational Linguistics (COLING 2002)*. Taipei, Taiwan.

Neumann, Günter, Christian Braun, and Jakub Piskorski. 2000. A divide-and-conquer strategy for shallow parsing of German free texts. In *Proceedings of ANLP-2000*, pages 239–246. Seattle, WA.

Schiehlen, Michael. 2003. Combining deep and shallow approaches in parsing German. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo.

Schiller, Anne, Simone Teufel, and Christine Thielen. 1995. Guidelines für das Tagging deutscher Textkorpora mit STTS. Tech. rep., Universität Stuttgart and Universität Tübingen.

Schmid, Helmut and Sabine Schulte im Walde. 2000. Robust German Noun Chunking with a Probabilistic Context-Free Grammar. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*, pages 726–732. Saarbrücken, Germany.

Trushkina, Julia and Erhard W. Hinrichs. 2004. A hybrid model for morpho-syntactic annotation of German with a large tagset. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 238–246. Barcelona, Spain.

Trushkina, Julia S. 2004. *Morpho-syntactic Annotation and Dependency Parsing of German*. Ph.D. thesis, University of Tübingen.

5

Solutions for Handling Non-concatenative Processes in Bantu Languages

ARVI HURSKAINEN

The paper discusses the description of non-concatenative processes in word formation by using examples from Bantu languages. The focal point is especially the verb, which may have up to fifteen morpheme slots. Because of space restrictions, only reduplication and non-cumulative morpheme concatenation will be discussed and solutions for implementation will be demonstrated. Some solutions require the use of such environments as provided by the Xerox tool package and Koskenniemi's Two-level morphology.

5.1 Introduction

Bantu languages display a number of features that cannot be effectively handled by using the basic finite state processing. Examples of such features include, for example, the full-stem reduplication of verbs, the non-cumulative sequence of verb morphemes, and the disjoining writing system.

Bantu languages exhibit a productive process of verb stem reduplication. Reduplicated forms cannot be described simply by adding reduplicated stems into the dictionary, because it is not the verb root but the full, possibly extended, stem of the verb that is reduplicated. When a verb may have up to 20 different extended stems, and each stem has at least three different surface forms, the listing of all of these forms in the lexicon is not practical.

Verb structures in Bantu languages are complex, comprising up to fifteen morpheme slots. There are a number of rules that restrict the co-occurrence

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

of certain morphemes. For the morphemes that precede the verb stem, it is possible to construct separate routes through the morpheme slots, allowing some, and disallowing others, from occurring in the sequence of morphemes. When the constraining morpheme is after the verb root, the problem cannot be solved by constructing separate routes, because this would require also the multiplication, perhaps several times, of the verb stems in the lexicon.

Some Bantu languages have adopted a writing system where verb morphemes, especially those preceding the stem, are written as separate words, while other languages, closely related to these, use a conjoining writing system. The problem in computational processing is how to keep these morphemes separate from similar morphemes which are not part of the verb and constitute grammatical words in themselves. Because of space restrictions, this problem is not discussed here.

The problems of implementing reduplication and the non-cumulative sequence of verb morphemes will be discussed below and solutions to them will be demonstrated. Some of the solutions have been implemented with general-purpose tools, and others require the use of such environments as provided by the Xerox tool package and Koskenniemi's Two-level morphology, licensed by Lingsoft.

Acknowledgements

I wish to thank Kimmo Koskenniemi for his advice in the early phases of developing the Swahili morphological parser in the 1980s, and Lauri Karttunen for introducing the solutions for solving the types of problems discussed here.

5.2 Reduplication

The extensive use of reduplication for grammatical and semantic purposes is a distinctive feature in most African languages. Part of reduplication is fairly easy to describe in the lexicon. Some types of reduplication, however, can hardly be described in a satisfactory way by simply concatenating morphemes. Below I shall describe two methods for handling reduplication.

5.2.1 Using basic finite state methods

In SALAMA, Swahili Language Manager (Hurskainen 2004b), reduplication was implemented by using the basic finite state concatenation. This led to a somewhat strange situation because, although the extended forms of verb stems were described with the help of continuation classes and corresponding sub-lexicons, each reduplicated verb stem had to be written in full. Thus the same basic verb may occur in the lexicon several times in different forms. Reduplicated forms were added to the lexicon if they occurred in texts. The verbs of the corpus of 15 million words have so far been included and new reduplicated forms are added when they are found in new texts. An example

of how the reduplicated forms of the verb `kata` (to cut) are described in this system is in (1).

(1)
```
Lexicon VRoot
katAkat       redup/V "katakat 'cut' SV SVO ";
katikAkatik   redup/V "katakat 'cut' SV STAT ";
katishAkatish redup/V "katakat 'cut' SVO CAUS ";
katizAkatiz   redup/V "katakat 'cut' SVO CAUS ";
katizwAkatizw redup/V "katakat 'cut' SV CAUS PASS ";
katwAkatw     redup/V "katakat 'cut' SV PASS ";
Lexicon Redup/V
A End;
A GenRel;
Lexicon GenRel
ye End;
```

Note that the use of two-level rules reduces the need of listing verb entries, because the rules handle the surface realisation of the verb-final A. Without the use of rules, each entry should be written three times, one for A > a as default, another for A > i in present tense negative, and another for A > e in subjunctive. The rules convert the A in the middle and at the end of the verb as required.

In addition to verbs, reduplication occurs frequently in pronouns and adverbs, and to a limited extent in adjectives. Reduplication has in these contexts mainly a semantic role, which has to be taken into consideration in a bilingual lexicon (Hurskainen 2004a). If reduplicated words, which do not inflect, are written as a single word without a space in between, they are easy to describe in the lexicon. There are, however, reduplicated adjectives with an alternating prefix defined by the noun class, written together as a single word. Some examples of the word `zuri` (good) are in (2).

(2)
```
mzurimzuri   End "zuri Adj 1/2-SG '  good '";
wazuriwazuri End "zuri Adj 1/2-PL '  good '";
mizurimizuri End "zuri Adj 3/4-PL '  good '";
kizurikizuri End "zuri Adj 7/8-SG '  good '";
zurizuri     End "zuri Adj 9/10-SG ' good '";
pazuripazuri End "zuri Adj 16-LOC '  good '";
```

A more elegant way of describing reduplicated adjectives is to formulate them as regular expressions. In this method, each of the basic stems is listed only once. The stem receives its prefixes from the sub-lexicon of prefixes, and the concatenated word is optionally reduplicated. This will be described in more detail in (11-12).

In sum, the method of describing the reduplicated stems directly in the lexicon has disadvantages, but also advantages. One disadvantage is that it is rather tedious to keep track of all real instantiations of reduplicated verbs. The method also requires continuing follow up because it is based on corpus occurrences and not on the grammatical word formation rules. On the positive side we can see the accuracy of the system because no potential verbs are there which do not occur in text. This saves the system from testing unnecessary paths, and also eliminates the risk of unnecessary additional ambiguity.

5.2.2 Solution based on regular expressions

It is well known that if a string or a sequence of strings can be expressed in the form of a regular expression, it can be repeated. Limited reduplication, repetition of strings, can be achieved even with two-level rules, although the power of this method is not sufficient for handling full scale applications such as Swahili and other Bantu languages.

The Xerox tool package contains a compile-replace algorithm, which makes it possible to include finite state operations other than concatenation into the morphotactic description (Beesley and Karttunen 2003: 379-380). In this method of describing non-concatenative phenomena, the initial lexical description is made by concatenating partial strings, usually morphemes, into well-formed words through a finite state lexicon structure. This partly abstract lexical description is mapped to the surface strings by applying morphophonological alternation rules. While in the usual description, following the terminology of Xerox, the lower language represents the orthographically correct word forms, in the compile-replace algorithm the initial network (i.e. the composition of the lexicon and the rules) is left abstract for including meta-morphotactic descriptions of non-concatenative phenomena.

When processing this kind of description, the morphophonological rules and lexicon, which are in the form of regular expressions, are first read and composed into a network. This network contains strings which also include meta-morphotactic descriptions in the form of regular expressions. The compile-replace command is applied to the lower side of the initial network, where it finds the meta-morphotactic descriptions, compiles them as regular expressions and replaces them in the lexicon network with the new network resulting from the compilation (Beesley and Karttunen 2003: 381-382).

The example in (3) illuminates how the above description is implemented with the Swahili verb `sema` (to say).

(3)
```
Multichar_Symbols
^[  ^]  @U.GENREL.abs@  @U.GENREL.pres@
@U.OBJ.abs@  @U.OBJ.pres@
Lexicon Root
Pref0@;   Pref1;   AdjStart;
```

```
Lexicon Pref0@
@U.GENREL.abs@   Pref0;
Lexicon Pref0
ha=Neg+:ha^   VStart;
ha=Neg+:ha^   Pref4;
a=Subjn+:a!   VStart;
a=Sbjn+:a!    Pref4;
Lexicon Pref1
< {a=Sp+}:{a} “@U.GENREL.abs@” >   Pref2@;
a=Sp+:a                            Pref4;
a=Sp+:a                            VStart;
Lexicon Pref2@
@U.GENREL.abs@ Pref2;
Lexicon Pref2
na=Pres+:na   VStart;
na=Pres+:na   Pref3@;
na=Pres+:na   Pref4;
Lexicon Pref3@
“@U.GENREL.abs@”   Pref3;
Lexicon Pref3
ye=1/2-Sg-Rel+:ye   VStart;
ye=1/2-Sg-Rel+:ye   Pref4;
Lexicon Pref4
< {ki=7/8-Sg-Obj+}:{ki} “@U.OBJ.pres@” >   VStart;
Lexicon VStart
@:@^[{   VStem;
Lexicon VStem
sem   VSuff;
Lexicon VSuff
+esh=Caus:Ish   VSuff2;
+esh=Caus:Ish   VFinV ;
VSuff2; VFinV;
Lexicon VSuff2
< {+w=Pass}:{w} “@U.OBJ.abs@” >   VFinV;
Lexicon VFinV
+a:A         EndSimple;
+a=Redup:A   EndRedup;
Lexicon EndSimple
0:}^1^]   End;
0:}^1^]   GenRel;
Lexicon EndRedup
0:}^2^]   #;
0:}^2^]   GenRel;
Lexicon GenRel
< {+ye=1/2-Sg-GenRel}:{ye} “@U.GENREL.pres@” > #;
```

```
Lexicon AdjStart
0:0^[{   AdjPref;
Lexicon AdjPref
m=1/2-SG+:m     AdjRoot;
wa=1/2-PL+:wa   AdjRoot;
mi=3/4-PL+:mi   AdjRoot;
ki=1/2-SG+:ki   AdjRoot;
0=9/10-SG+:0    AdjRoot;
pa=16-LOC+:pa   AdjRoot;
Lexicon AdjRoot
zuri   EndSimple;
zuri   EndRedup;
```

In the lexicon above, the upper-side language is represented in such a way that it contains a sequence of lexical morphemes and their grammatical flags, and morpheme boundaries are shown with a plus sign. The lower-side language is also abstract in that it contains characters in upper case that are subject to alternation rules for producing correct surface forms. Particularly important in the lower-side language is the section of the string that is subject to reduplication. This section is delimited with special multi-character symbols `^[` and `^]`. Whatever is between these symbols is a regular expression that can be manipulated accordingly, in this case, repeated. We also see that the actual string to be defined as a regular expression is enclosed with curly brackets `{` and `}` for making sure that the string is interpreted as a regular expression. In (4) is an example of how the surface string `anasemeshasemesha` (he makes to speak) is represented in the lexicon. Note that the influence of alternation rules is here excluded.

(4)
```
upper: a=1/2-Sg-Sp+na=Pr+sem+esh=Caus+a=Redup
lower: a           na@^[{sem Ish      A}^2^]
```

The multi-character symbol `^2` in the lower string stands for repeating, i.e. the preceding regular expression enclosed between curly brackets `{` and `}` is repeated. The alternation rules rewrite the I and A as needed in the surface string. In (5) we show in stages how the final network is compiled by using a script file.

(5)
```
xfst -e "read regex < rules.txt"
-e "read lexc < redup.lex"
-e "compose"
-e "compile-replace lower"
-e "substitute symbol 0 for Caret"
-e "substitute symbol 0 for !"
-e "substitute symbol 0 for @"
-e "save redup.fst"
-stop
```

Note that all diacritics needed as triggers in alternation rules are deleted in the final network. They are substituted with the zero symbol. The command sequence in (6) shows how the network operates.

(6)
```
xfst[0]: load redup.fst
Opening 'redup.fst'
Closing 'redup.fst'
xfst[1]: up anasema
a=1/2-Sg-Sp+na=Pr+sem+a
xfst[1]: up anasemasema
a=1/2-Sg-Sp+na=Pr+sem+a=Redup
```

We see that both the simple and reduplicated stems are analysed. The simple stem is analysed when the repetition trigger is set to `^1`. We can also test the network in the other direction, as shown in (7).

(7)
```
xfst[1]: down a=1/2-Sg-Sp+na=Pr+sem+a
anasema
xfst[1]: down a=1/2-Sg-Sp+na=Pr+sem+a=Redup
anasemasema
```

As is shown in (8), adding verb affixes does not affect the correct realisation of the verb stem.

(8)
```
xfst[1]: up anasemesha
a=1/2-Sg-Sp+na=Pr+sem+esh=Caus+a
xfst[1]: up anasemeshasemesha
a=1/2-Sg-Sp+na=Pr+sem+esh=Caus+a=Redup
xfst[1]: up anayesemeshasemesha
a=1/2-Sg-Sp+na=Pr+ye=1/2-Sg-Rel+sem+esh=Caus+a=Redup
xfst[1]: up anakisemeshasemesha
a=1/2-Sg-Sp+na=Pr+ki=7/8-Sg-Obj+sem+esh=Caus+a=Redup
```

The lexicon in (3) shows that the verb stem is not always the last element in the verb. For example, the marker of the general relative is attached to the end of the verb stem, and this suffix is not reduplicated. The formalism also handles such cases, as shown in (9). Ungrammatical concatenations cause a failure.

(9)
```
xfst[1]: up asemaye
a=1/2-Sg-Sp+sem+a+ye=1/2-Sg-GenRel
xfst[1]: up asemasemaye
a=1/2-Sg-Sp+sem+a=Redup+ye=1/2-Sg-GenRel
xfst[1]: up akisemasemaye
a=1/2-Sg-Sp+ki=7/8-Sg-Obj+sem+a=Redup+ye=1/2-Sg-GenRel
xfst[1]: up anasemasemaye
```

The negative present and subjunctive affect the verb-final vowel, and this

is implemented with alternation rules as shown in (10). The analysis fails if the final vowel is not correct.

(10)
```
xfst[1]: up hasemi
ha=Neg1/2-SG-SP+sem+a
xfst[1]: up hasemisemi
ha=Neg1/2-SG-SP+sem+a=Redup
xfst[1]: up aseme
a=Sbjn1/2-SG-SP+sem+a
xfst[1]: up asemeseme
a=Sbjn1/2-SG-SP+sem+a=Redup
xfst[1]: up hasema
xfst[1]: up hasemasema
xfst[1]: up haseme
```

5.2.3 Reduplicated adjectives

We saw in (2) that inflecting reduplicated adjectives require multiple listing in the dictionary if only the basic concatenation method in the system is available. Here we show that this can be avoided by describing the adjective, together with its prefix, as a regular expression. In the example lexicon (3) the solution for adjectives is also demonstrated. The adjective `zuri` (good) is described, together with a sample of alternative prefixes. Test examples in (11) show that both the simple and reduplicated forms are recognised and analysed accordingly.

(11)
```
xfst[1]: up mzuri
m=1/2-SG+zuri
xfst[1]: up mzurimzuri
m=1/2-SG+zuri=Redup
xfst[1]: up wazuriwazuri
wa=1/2-PL+zuri=Redup
xfst[1]: up pazuri
pa=16-LOC+zuri
xfst[1]: up pazuripazuri
pa=16-LOC+zuri=Redup
```

When the upper-side language is applied to the lower-side language, we get the correct surface forms. When a string with different prefixes in the first and second part of the reduplicated stem is entered, the test fails.

(12)
```
xfst[1]: down m=1/2-SG+zuri=Redup
mzurimzuri
xfst[1]: down pa=16-LOC+zuri=Redup
pazuripazuri
xfst[1]: up kizurizuri
xfst[1]: up zurikizuri
xfst[1]: up kizuripazuri
```

5.3 Non-concatenative dependencies

In the lexicon in (3) there are so-called flag diacritics, the purpose of which is to constrain the occurrence of incompatible features in the same string. The relative marker after the verb stem blocks the occurrence of the relative marker in Pref3, and also a number of other prefixes. Another similar case is that the object prefix cannot co-occur with the passive marker. The triggers for both types of constraints are located on different sides of the verb stem, and this calls for the use of flag diacritics (Beesley and Karttunen 2003: 339-373). The unification flag diacritics are used in the lexicon for preventing the co-occurrence of unwanted features in the same string. In the current example lexicon (3), the flag diacritics are made visible on the upper and lower side of the transducer, so that they function correctly in analysis and production.

Another possibility for constraining the unwanted combination of morphemes here would be to use pairs of P-type (positive) and R-type (require) diacritics for defining the correct strings, where both of the types of the same flag with the same value must co-occur. Space does not allow the demonstration of this alternative. Examples in (13) show how the constraints with the U-type (unification) flag diacritics work.

(13)
```
xfst[1]: up anayesema
a=1/2-Sg-Sp+na=Pr+ye=1/2-Sg-Rel+sem+a
xfst[1]: up asemaye
a=1/2-Sg-Sp+sem+a+ye=1/2-Sg-GenRel
xfst[1]: up anayesemasema
a=1/2-Sg-Sp+na=Pr+ye=1/2-Sg-Rel+sem+a=Redup
xfst[1]: up asemasemaye
a=1/2-Sg-Sp+sem+a+Redup+ye=1/2-Sg-GenRel
xfst[1]: up anayesemaye
xfst[1]: up asemayesemaye
```

Note that if the verb stem with the general relative suffix is reduplicated, the analysis fails. Only the verb stem, simple or extended, is reduplicated, and the relative marker is attached to the end of the reduplicated stem.

5.4 Conclusion

We have discussed non-concatenative processes that take place in Bantu languages on the word level and tested methods for solving them. Our conclusion on the basis of the tests is that the environment offered by the Xerox tool package offers elegant solutions to the problems discussed. The reduplication of verbs and adjectives and constraining the co-occurrence of non-contiguous morphemes can be described simultaneously and compiled into a network.

Reduplication can be handled also in the basic finite state lexicon, but this is more labor-intensive than with Xerox tools. The number of various redupli-

cated extended verb stems is in practice much more limited than the number of extended non-duplicated verb stems. The number of extra listings required by reduplicated verbs in Swahili for handling normal text is less than 300. The reduplicated adjectives can be listed as such if they occur in real language.

References

Beesley, Kenneth and Karttunen, Lauri. 2003. *Finite State Morphology*. Series: CSLI Studies in Computational Linguistics. Stanford: Center for the Study of Language and Information.

Hurskainen A. 2004a. Optimizing Disambiguation in Swahili. In *Proceedings of COLING-04, The 20th International Conference on Computational Linguistics, Geneva 23-27.8. 2004.* Pp. 254-260.

Hurskainen, A. 2004b. Swahili Language Manager: A Storehouse for Developing Multiple Computational Applications. *Nordic Journal of African Studies, 13(3): 363-397.* `www.njas.helsinki.fi`

Koskenniemi, Kimmo 1983. *Two-level morphology: A general computational model for word-form recognition and production.* Publications No.11. Department of General Linguistics, University of Helsinki.

6

A Method for Tokenizing Text

RONALD M. KAPLAN

6.1 Introduction

The stream of characters in a natural language text must be broken up into distinct meaningful units (or tokens) before any language processing beyond the character level can be performed. If languages were perfectly punctuated, this would be a trivial thing to do: a simple program could separate the text into word and punctuation tokens simply by breaking it up at white-space and punctuation marks. But real languages are not perfectly punctuated, and the situation is always more complicated. Even in a well (but not perfectly) punctuated language like English, there are cases where the correct tokenization cannot be determined simply by knowing the classification of individual characters, and even cases where several distinct tokenizations are possible. For example, the English string *chap*. can be taken as either an abbreviation for the word *chapter* or as the word *chap* appearing at the end of a sentence, and *Jan*. can be regarded either as an abbreviation for *January* or as a sentence-final proper name. The period should be part of the word-token in the first cases but taken as a separate token of the string in the second. As another example, white-space is a fairly reliable indicator of an English token boundary, but there are some multi-component words in English that include white-space as internal characters (e.g. *to and fro*, *jack rabbit*, *General Motors*, *a priori*).

These difficulties for English are relatively limited and text-processing applications often either ignore them (e.g., simply forget about abbreviations and multi-component words—there are many more difficult problems to worry about) or treat them with special-purpose machinery. But this is a much bigger problem for other languages (e.g. Chinese text is very poorly

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

punctuated; the Greek on the Rosetta stone has no spaces at all) and they require a more general solution.

This paper describes just such a general solution to the problem. We characterize the tokenizing patterns of a language in terms of a regular relation T that maps any tokenized-text, a string of characters with token-boundaries explicitly marked, into the concrete string of characters and punctuations marks that would be an expression in proper typography for the given sequence of tokens. For example, it maps the tokenized-text

(1) Happily TB , TB he TB saw TB the TB jack rabbit TB in TB Jan. TB .

into the actual text

(2) Happily, he saw the jack rabbit in Jan.

where TB is the explicit token-boundary marker. However, (1) is not the only tokenized-text that can be expressed concretely as (2). Alternative sources include texts in which there is a token-boundary between *jack* and *rabbit* (3a) and where the last word is taken as a proper name (3b).

(3) a. Happily TB , TB he TB saw TB the TB jack TB rabbit TB in TB Jan.
b. Happily TB , TB he TB saw TB the TB jack rabbit TB in TB Jan TB.

If T maps these (and perhaps other) alternative sources into the concrete text in (2), then its inverse T^{-1} will map the text in (2) back to the different explicitly tokenized sources in (1) and (3).

The correct tokenizing patterns for a language are thus defined by a regular relation or finite-state transducer, formal devices that have the power to characterize the complexity and ambiguity of punctuation conventions across the languages of the world. We describe a particular algorithm for applying such a transducer to a given text. This algorithm is a variant of the general composition algorithm for finite-state transducers, but it is specialized to the particular properties of text streams: they are usually quite long but they can be represented by finite-state machines with a single (acyclic) path. The algorithm uses this fact to provide efficient management of temporary storage and to provide guaranteed output for individual substrings of the text as rapidly as possible. The output is guaranteed in the sense that no data later in the text will cause the tokenization of a previous substring to be retracted as a possible tokenization—unless the text as a whole is ill-formed. Thus a client can safely invest its own computational resources for higher-order applications (text indexing, question-answering, machine translation...) without fear of wasting effort on incremental tokenizations that have no valid future.

6.2 Tokenizing Relations

A tokenizing relation can be defined for a particular language by a set of rules that denote regular relations (Kaplan and Kay, 1994), by a regular expression over pairs, or by the state-transition diagram of a finite-state transducer. For example, the following ordered set of obligatory context-sensitive rewriting rules defines a regular relation that gives a first-approximation account of English punctuation conventions:

(4) a. period TB $\rightarrow \varepsilon$ / __ period
b. TB $\rightarrow \varepsilon$ / __ right-punctuation
c. TB $\rightarrow \varepsilon$ / left-punctuation __
d. TB $\rightarrow$ space
e. space $\rightarrow$ white-space$^+$

The first rule deletes an abbreviatory (word-internal) period when it comes before a sentence-final period, as needed in the mapping of (1) to (2) above. The second rule causes token-boundaries to be deleted in front of punctuation marks that normally appear at the ends of words (e.g. closing quotes, commas) while the third deletes token boundaries after opening punctuation marks (e.g. open quotes). The fourth converts any remaing token-boundaries to space characters, and the fifth expands those spaces, as well as the internal spaces of multi-component tokens, to arbitrary sequences of white-space characters (space, carriage return). Other rules could be added to deal, for example, with optional end-of-line hyphenation, contractions, and sentence-initial capitalization. These rules are written in the generative direction, which may seem counterintuitive since the problem at hand is a recognition problem. But we have found that, as a general principle, such mappings are easier to characterize as reductions of more structured (i.e. explicitly tokenized) to less structured representations, with recognizers obtained by applying the relations in reverse. Grammars of this type can be compiled into finite-state transducers using the methods described by Kaplan and Kay (1994). Systems of two-level transducers (Koskenniemi, 1983) or two-level rules (Kaplan and Kay, 1994) can also be used to define regular tokenizing relations.

6.3 The General Idea

With such a relation in hand, the problem of finding the set of all admissible tokenizations of a text is simply an instance of the general problem of recognizing a text with respect to a transducer. If the text is interpreted as a (single-string) regular language, the solution to this problem is computed by the recognition operator Rec, defined as

(5) $\mathrm{Rec}(T, \mathit{text}) \stackrel{\mathrm{def}}{=} \mathrm{Dom}(T \circ \mathrm{Id}(\mathit{text}))$

This operator constructs the identity relation that takes the given text into itself, and composes that with the tokenizing relation. This has the effect of restricting the general tokenizer T to the subrelation that only has the given text as its output side. The domain of that restricted relation is exactly the set of tokenized strings that T would express as the given text (see Kaplan and Kay, 1994, for a discussion of the domain and identity relations and other relevant concepts). The result is a regular set (perhaps an infinite set if T is not linear bounded) that can be represented as a finite-state automaton. This formula is equivalent to Sproat's (1995) characterization of the tokenizing problem, modulo a transposition of the relational coordinates.

Formula (5) defines the computation we want to perform, and it is made up of operations (identity, composition, domain-extraction) that are easy to implement given a finite-state transducer presentation of T and a finite-state machine representation of the text. But the normal implementations of these operations would be quite impractical when applied to a long text. They tend to build intermediate data structures that are linear in the length of the text, and they would produce no results at all until the entire text has been processed. The standard algorithms may be acceptable if we are willing to preprocess a long text to break it up into sentence-size chunks and then operate on those one at a time. This is the arrangement contemplated by Sproat (1995), for example, but it requires additional heuristic machinery and may not deal gracefully with sentence-boundary ambiguities.

We describe here a method of evaluating this formula that is general and uniform and applies with practical efficiency to texts of arbitrary length. The method has four desirable properties:

1. It produces output incrementally, and does so whenever it reaches a point in the text where all local tokenization ambiguities have been resolved (so-called "pinch-points").
2. The temporary storage it requires is bounded by the maximum distance between pinch-points. Storage can be recycled in the computation between pinch-points when any path of ambiguity dies out.
3. It never causes previously produced output to be retracted, unless the text as a whole has no proper tokenization (is globally ill-formed).
4. It combines nicely with higher-level sources of lexical information, so that dictionary constraints on tokenization (e.g. a list of known abbreviations) can be taken into account without modifying the run-time algorithm. It can also be combined with syntactic constraints defined in grammatical formalisms that are closed under composition with regular relations. These include context-free grammars and Lexical Functional

Grammars (Kaplan and Bresnan, 1982).[1]

In a well-punctuated language, pinch-points come fairly close together (e.g. almost (but not always) at every punctuation mark). Thus, this algorithm is practical even for relatively simple languages like English, and it obviates the need to develop special-purpose code to take advantage of its limited tokenization patterns (or to provide less than accurate approximations). In a poorly punctuated language the pinch-points will not be as close together, so that reliable output will be provided less frequently and the amount of internal storage will be larger. But these effects are directly proportional to the inherent complexity of the language—it is extremely unlikely that there is a simpler way of handling the facts of the matter.

6.4 A Practical Method

We start from the general implementation of the composition, domain, and identity operations in the formula

$$\mathrm{Dom}(T \circ \mathrm{Id}(text))$$

This sort of formula is usually evaluated by operating on the finite-state transducer accepting T and the finite-state machine accepting the text. The composition algorithm is normally a quadratic operation bounded by the product of the number of states of its two operand transducers (in this case representing T and $\mathrm{Id}(text)$. It would produce an output transducer with states corresponding to pairs of states of T and $\mathrm{Id}(text)$, with the transitions from each of the pair-states encoding what the next moves are, given that that the particular pair of states in the T and $\mathrm{Id}(text)$ fsts has been reached. An indexing structure must be maintained, so that the data-structure representing a given pair-state can be obtained if that same pair of states is reached on alternative paths through T and $\mathrm{Id}(text)$. In this event, the forward moves previously determined for that pair-state are valid on the new path, and the alternatives can be merged together. This can happen when the composition of disjunctive and infinite relations (with cyclic fsts) is computed. Indeed, if the merger is not performed, then the composition computation may not terminate on cyclic input machines. This general algorithm is impractical, or at least unattractive, for tokenizing an arbitrary text because it would require an amount of indexing storage proportional to the length of the text.

[1] LFG's nonbranching dominance (off-line parsability) condition must be generalized slightly to preserve decidability of the membership problem in the case that the tokenizing relation T is not linear-bounded. A dominance chain must be regarded as nonbranching if a re-encountered category is paired not just with the same position of a single string but with the same suffix language of a tokenized regular set (as denoted by a state of the finite-state machine that represents the tokenization result).

Our tokenizing method circumvents these difficulties by finding correlations between segments of the text and segments of the tokenizing relation. We let T stand interchangeably for the regular relation or a transducer that represents it, as appropriate. We suppose that s is the start-state of T and, without loss of generality, that f is its single final state. If q and r are states of T, we define ${}_qT_r$ to be the set of all pairs of strings for which there is an accepting path in T that starts at state q and ends at state r. We also interpret the text itself as a finite-state automaton with numbered junctures between characters playing the role of its states. Thus ${}_i text_j$ is the substring of the text between positions i and j and the entire text of n characters is denoted as ${}_0 text_n$. We now say that

(6) A text-position k is a *pinch-point for* $\mathrm{Rec}(T, text)$ *with pinch-state* p iff

$$\mathrm{Rec}(t, text) = \mathrm{Rec}({}_sT_p, {}_0 text_k) \cdot \mathrm{Rec}({}_pT_f, {}_k text_n).$$

The raised dot denotes the usual concatenation operator for formal languages: the concatenation $L_1 \cdot L_2$ is the language $\{xy \mid x \in L_1, y \in L_2\}$. Thus, if k is a pinch-point and p is its pinch-state, any tokenization *t* of the text can be partitioned into two strings t_1 and t_2 such that t_1 is a tokenization of the subtext before position k provided by a path in T up to state p and t_2 is a tokenization for the rest of the text provided by a path in T leading from state p. If such a k-p pair can be identified, then the strings in the regular language $\mathrm{Rec}({}_sT_p, {}_0 text_k)$ can be supplied as incremental output to a client application, and the legal continuations of all those strings are completely determined by k and p and can be computed by evaluating the suffix tokenization-expression $\mathrm{Rec}({}_pT_f, {}_k text_n)$. The condition in (6) can be applied iteratively, so that a sequence of incremental outputs can be provided one after the other.

We see that the computation of $\mathrm{Rec}(T, text)$ can be suspended whenever a pinch-point is reached and that all tokenizations for the subtext to that point can be delivered as output. Moreover, the detailed pair-state indexing structures required by the composition operator can be discarded at such a point, and only two pieces of information, the position k and the pinch-state p, are needed to continue tokenizing the rest of the text. The challenge, of course, is to identify pinch-points and pinch-states at the earliest positions of the text; that is what our method for tokenizing text is organized to do.

We observe that the text itself is a linear string and so $\mathrm{Id}(text)$ is represented by a single-path transducer. This means that a composition path that has advanced beyond a particular text state/position will never return to that same text-position. Thus, the indexing structures necessary to find previous pair-states and carry out future mergers do not have to be maintained across

the whole text. In particular, if we carry out the state-traversal computation of the composition algorithm in a breadth-first manner, then we can recycle the storage for all the indexing structures involving a given text-position as soon as we have examined all possible next-moves at that position.

The next obvious observation is that we can simulate the identity map on the text by letting each character stand for its own image on the other tape of the (now fictional) identity transducer. We can implement the breadth-first composition in this special case by simply maintaining a list of configurations at a current character position. These configurations represent all the states in the transducer T that were paired up with that text position, and they record for eventual output the domain characters that were produced along the way.

In pursuing the future of a given configuration, we compare the transitions leaving its transducer state with the next character in the text. We build new configurations at the next-character position for those transitions that match the next text character. If a transition has an ε (empty string) so that it does not advance through the text, the new configuration is added to the end of the list of current-position configurations. However, before we add a new configuration to one of the lists, we check to see whether a configuration with the same transducer-state is already present. If such a configuration does exist, we add alternatives to its output record instead of adding a separate new configuration. This has the same effect as the merger step in the general composition algorithm: it collapses alternative paths and insures termination even if the T transducer is cyclic.

Thus, we maintain two lists of configurations: one for the current text position, one for the next. When we have exhaustively processed all the current position's configurations, we can recycle them to a free-list, move forward to the next position, and start processing the configurations there.

The output of this computation is given by the domain side of the T transitions that carry us across the text. We construct incrementally the transitions of the finite-state machine that represents this output by associating an output-state with each of the configurations. The transitions of this state are labeled with the domain transitions of matching arcs of T. A subtlety, however, is that these transitions point backwards, towards the output state of the configuration currently providing for the match against the text. We also maintain on each output-state a reference count of the number of (backwards) transitions that point to it. This enables us to implement a reference-counting garbage collector that collects the storage for failed paths as soon as possible. A path fails if there are no arcs at the T state of a configuration that match the current text-character. In that case, the (backward) transitions at the configuration's output state are collected, and the reference counts of the states they point to are decremented. If those counts go to zero, then those states and their backward-pointing arcs can also be reclaimed, recursively.

This arrangement will collect all unneeded output storage as soon as possible, provided that T itself is linear-bounded. If T is linear-bounded, then $\mathrm{Dom}(T \circ \mathrm{Id}(text))$ will be finite and there will be no loops in the output finite-state machine. Otherwise, there can be infinitely many tokenized strings, represented by a cyclic fsm, and there can be self-justifying loops that will not be incrementally collected when a cyclic path cannot later be extended. In that case, the loop-structures on failed paths will not be reclaimed until a pinch-point is reached, the current output fsm is provided back to the client, and all current-output structures are freed.

We note as an aside that a non-linear-bounded tokenizing relation producing infinitely many results is not entirely implausible. It is a convention of English text that appositives and nonrestrictive relative clauses are both set off by commas, and the appearance of commas is thus an important signal for the higher-level syntactic analysis of a text. Logically then, a nonrestrictive relative ending in an appositive should be followed by two commas, but a sequence of commas is always reduced to one by the rules of English typography. This reduction can be expressed by adding the following rewriting rule to the specification of T:

(7) comma TB $\rightarrow \varepsilon$ / __ comma

The resulting tokenizer will introduce an arbitrary number of tokenization commas in front of every comma that appears in the actual text, leaving it to a syntactic analyzer to pick the ones that satisfy grammatical constraints. This may not be helpful for unsophisticated uses of the output, but it can enable substantial simplifications of grammars for applications requiring a full syntactic or semantic analysis.

To finish describing our tokenizing method, we observe that there is an easy way of recognizing that a pinch-point has been reached. This is a point at which all open paths have come together at a single state of T, so that there is only one way of proceeding further ahead. This is marked by the fact that there is only one item on the list of current configurations. Several items are on the list if there are current ambiguities, one item remains when those ambiguities have a determinate future, and the list becomes empty only when the text itself is ill-formed. When there is only a single open configuration, the output trailing behind that configuration can be provided to the client, with the guarantee that the current output will not be retracted unless the text as a whole is ill-formed (this can happen only if the range of T is not Σ^*; it is decidable in advance whether T has this unattractive property). Thus, it is safe to provide output whenever a singleton configuration-list is reached. It may be desirable not to produce output in every such situation, but instead let the output accumulate internally until a token boundary is also reached. This means that the client will receive outputs only in chunks of meaningful

units. In a well-punctuated language, the output will come essentially at every word boundary. The pinch-state needed for tokenizing the rest of the text is the state of T stored in the single configuration.

The output can be produced by reversing the arcs of the output-fsm that has been threaded through the configuration, and then providing the result as an fsm structure to the client. Or the output-fsm can be copied to some other graph data structure as determined by the client, for example, the initial chart of a chart-parser for higher-level syntactic analysis.

6.5 Higher-level Lexical Constraints

The tokenizing relation T can be modified in various ways to incorporate higher-level constraints on tokenization and thus to eliminate inappropriate tokenizations at an early stage of processing. If these additional constraints can be expressed as regular relations or regular languages, they can be composed into the original formula to provide a more restrictive tokenization mapping. For example, suppose that the lexical relation L is regular and maps sequences of word stem-and-tag-strings separated by token boundaries into the corresponding inflected forms of the words (see, for example, Karttunen *et al.*, 1992). Then

(8) $\mathrm{Dom}(L \circ T \circ \mathrm{Id}(\mathit{text}))$

defines the morphological analysis of the tokenized text. Composition is an associative operation, so we can rewrite this as

(9) $\mathrm{Dom}((L \circ T) \circ \mathrm{Id}(\mathit{text}))$

Since L and T are independent of the text, the composition $L \circ T$ can be evaluated at compile-time to produce a new transducer LT that maps tokenized-stems to surface text. Applying our algorithm to LT will produce the tokenized morphological analysis of the text.

If a full lexical or morphological relation is not available or not desired, it may still be advantageous to augment the tokenizing relation with a small amount of additional information. This can be used to control the rampant ambiguity that comes from letting every space be construed as expressing either a token boundary or the internal space of a multi-component word, and from letting every period be interpreted as marking either a sentence boundary or an abbreviation. Suppose that M is a list of all the multi-component words that are to be allowed, that A is a list of known abbreviations, and that X is the regular set of all strings not containing either space or period. Define the regular language XAM to be the union of X, A, and M. The strings in this language do not contain spaces or periods unless they are in M or A. We construct $[(XAM\ \mathrm{TB})^* XAM]$, allowing for the elements of XAM to be packed together with token boundary separators. The relation

(10) $\mathrm{Id}([(XAM\ \textsc{tb})^* XAM]) \circ T$

can then be used as a tokenizing relation that produces space/period ambiguities only for the strings contained in M or A.

6.6 The Best Tokenization

Even with higher-level lexical constraints there may be many more alternative tokenizations than are desired, and many of them may be quite improbable. Sproat (1995) suggests a different approach to selecting the most appropriate outputs. He proposes to characterize the tokenization relation as a weighted regular relation, where each mapping is associated with a weight, perhaps a probability, that measures its degree of acceptability. The idea is to provide the output produced along the best, most highly weighted path through the transducer. It is easy to accommodate this proposal within our pinch-point framework, since the best-path operator BP distributes over concatenation (11) and thus distributes over our pinch-points (12):

(11) $\mathrm{BP}(L_1 \cdot L_2) = \mathrm{BP}(L_1) \cdot \mathrm{BP}(L_2)$

(12) $\mathrm{BP}(\mathrm{Rec}(t, text)) = \mathrm{BP}(\mathrm{Rec}({}_sT_p, {}_0text_k)) \cdot \mathrm{BP}(\mathrm{Rec}({}_pT_f, {}_k text_n))$

We simply provide at each pinch-point the output corresponding to the best path rather than the full set of tokenizations. This will produce the best tokenization of the entire text.

References

Kaplan, R. and Bresnan, J. 1982. Lexical Functional Grammar: A formal system for grammatical representation. In J. Bresnan (ed.), *The mental representation of grammatical relations.* Cambridge: MIT Press, 173-281.

Kaplan R. and Kay, M. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3), 331-378.

Karttunen, L., Kaplan, R., and Zaenen, A. 1992. Two-level morphology with composition. In *COLING'92*, Nantes, France, 141-148

Koskenniemi, K. 1983. *Two-level morphology: A general computational model for word-form recognition and production.* Publication 11, University of Helsinki, Department of General Linguistics, Helsinki.

Sproat, R. 1995. A finite-state architecture for tokenization and grapheme-to-phoneme conversion for multilingual text analysis. In S.Armstrong and E. Tzoukermann (eds.), *Proceedings of the EACL SIGDAT Workshop*, Dublin, Ireland, 65-72.

7

Phonotactic Complexity of Finnish Nouns

FRED KARLSSON

7.1 Introduction

In the continuous list of publications on his homepage, Kimmo Koskenniemi gives an item from 1979 as the first one. But this is not strictly speaking his first publication. Here I shall elevate from international oblivion a report of Kimmo's from 1978 from which the following introductory prophecy is taken: "The computer might be an extremely useful tool for linguistic research. It is fast and precise and capable of treating even large materials" (Koskenniemi 1978: 5).

This published report is actually a printed version of Kimmo's Master's Thesis in general linguistics where he theoretically analyzed the possibilities of automatic lemmatization of Finnish texts, including a formalization of Finnish inflectional morphology. On the final pages of the report he estimates that the production rules he formulated may be formalized as analytic algorithms in several ways, that the machine lexicon might consist of some 200,000 (more or less truncated) stems, that there are some 4,000 inflectional elements, that all of these stems and elements can be accommodated on one magnetic tape or in direct-access memory, and that real-time computation could be 'very reasonable' (*varsin kohtuullista*) if the data were well organized and a reasonably big computer were available (ibid.: 52-53). I obviously am the happy owner of a bibliographical rarity because Kimmo's dedication of 1979 tells me that this is the next to the last copy.

This was five years before two-level morphology was launched in 1983 when Kimmo substantiated his 1978 exploratory work by presenting a full-blown theory of computational morphology and entered the international computational-linguistic scene where he has been a main character ever since.

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

In the 1970s Kimmo worked at the Computing Centre of the University of Helsinki but he also studied general linguistics and engaged himself in linguistic computing, i.e. the computational treatment of corpora for purposes of linguistic research. When our collaboration started in 1980, he had obtained a copy of the magnetic tape of the Reverse Dictionary of Modern Standard Finnish (Tuomi 1972) of which he made various refined machine versions that were of great importance for our early theoretical and descriptive work in computational morphology.

My book *Suomen kielen äänne- ja muotorakenne* (Structure of Finnish Phonology and Morphology, Karlsson 1983) profited greatly from the computerized lexical and other corpora provided by Kimmo. In commemoration of Kimmo's work in linguistic computing in those early days I shall here present some observations on the phonotactic complexity of Finnish nouns using those same valuable data from around 1980 out of which not all potential scholarly juice has yet been squeezed.

If phonemically /long/ vowels and consonants are treated as combinations of identical phonemes (the standard solution), Finnish has eight vowel and thirteen consonant phonemes, /i e æ y œ u o a/ and /p t k d s h v j l r m n ŋ / for which I henceforth am going to use their standard orthographic equivalents <i e ä y ö u o a p t k d s h v j l r m n ng> (/ŋ/ is phonemic only when long, symbolized by the digraph <ng>).

Morpheme boundaries are indicated by '+', syllable boundaries by the period, '.'.

7.2 Number of nouns

How many nouns are there in Finnish (or in any language)? The question might seem silly because nouns are the prime example of an open-ended word class. But the question is relevant if rephrased to concern either (i) the size of the core vocabulary, i.e. the 'central words' presumed to be known by every normal native speaker, or (ii) the number of atomic free root morphemes, to the exclusion of derivatives and compounds. Here, I shall try to answer (ii) on the basis of the material in the *Reverse Dictionary of Modern Standard Finnish* (RDF; Tuomi 1972).

The starting point of RDF was the data comprising the original version of the standard Finnish reference dictionary *Nykysuomen sanakirja* (NS; 1952-1962) with 207,256 entries the majority of which were more or less productively formed compounds. RDF contains all basic words, derived words, basic components of compounds, compounds the basic parts of which occur only in compound words, and a handful of clitics (which are not words proper). In all, RDF comprises 72,711 entries which implies that the number of compounds in NS is 134,545. (The machine-readable version of RDF available

at the Department of General Linguistics, University of Helsinki, has 72,785 entries.)

Of the 72,785 entries in RDF 34,673 (47.6 %) have the code 'S', short for noun (including words like *suomalainen* 'Finn; Finnish (adj.)', which have homonymous nominal and adjectival readings). But among these there are huge numbers of fully productive derivatives like *pysäköi+nti* 'parking', *nuole+skel+u* '(habit of) licking', *ost+el+ija* 'one who habitually buys', *tilaa+ja* 'one who orders', *tanssi+ja+tar* 'female dancer', *dumppa+us* 'dumping', *suvaitse+vais+uus* 'tolerance', *riittä+mättöm+yys* 'insufficiency', *marksi+lainen* 'Marxist'. There are at least 16,000 – 17,000 fully transparent derivatives like these.

Among the remaining 18,000 – 19,000 nouns there are still many that are morphologically more or less complex. For example, there are 251 words ending in *-isti* and 285 ending in *-ismi* like *aktivisti* 'activist', *aktivismi* 'activism'. A conservative estimate is that 1,500 more nouns could be decomposed by careful morphological (and even etymological) analysis. Furthermore there are at least 5,000 clear borrowings, e.g. around 4,000 nouns containing the foreign letters <b d g z x f c š w q> (words with the sequence <ng> are not included among these, nor are the genuinely Finnish ones with <d>).

This would put the number of genuinely Finnish, morphologically atomic noun roots in the vicinity of 12,000, perhaps even lower, which I think is much less than popular beliefs would hold. The next section offers an analysis of the 18,500 nouns which contain no fully transparent productive derivatives.

7.3 Canonical patterns

Table 1 depicts the incidence of monosyllabic noun roots in the nominative singular case form that belong to the core vocabulary, i.e. words which are known to any normal speaker of Finnish. This means that e.g. musical terms such as *do, re, mi, es, ais, cis,* or obsolete and dialectal words and non-assimilated borrowings are not considered, e.g. *hie, huu, hyy, gnu, bei, boa, jen, yen, sir*. There are eight potential monosyllabic patterns:

The 29 monosyllabic nouns listed in Table 1 comprise just a fraction of one percent of the 12,000 atomic root nouns surmised above. The only significant monosyllabic nominal pattern is CVV. This is surprising because from the viewpoint of general markedness theory one would have predicted both that the phonotactically simpler CV-pattern would occur and that it would occur more frequently than CVV. But CV-nouns (and verbs) are effectively prohibited. The reason for the prevalence of CVV over CV cannot be morphological either because there are CV-pronouns that are regularly inflected: *tä+hän* 'into this one' (illative singular), *jo+hon* 'into which', *mi+hin* 'into which'.

Table 1. Monosyllabic noun patterns in Finnish.

Pattern	Number	Examples
V	-	
VV	1	yö
CV	-	
CVV	24	hai, hää-, jää, koi, kuu, kyy, luu, maa, pii, puu, pyy, pää, suo, suu, syy, sää, tee, tie, tiu, työ, täi,voi, vuo, vyö
CVC	-	
VC	-	
CVVC	4	mies, syys, hius, ruis
VVC	-	
sum	29	

The singleton VV-word *yö* 'night' stands out as an exception. Monosyllabic VVV-strings are prohibited; *aie* 'intention' is disyllabic, cf. *aikee+n* (genitive singular).

As for closed monosyllables, (C/V)VC-nouns are non-existing (*ien, oas, äes* are disyllabic). CVVC-nouns are extremely marginal, there are four of them, which all have marked inflection. *Syys* 'autumn' allows no inflection, the other three are all inflected in different ways: *mies* 'man'- *miehe+n*, *hius* 'hair' - *hiukse+n*, *ruis* 'rye' - *rukii+n*. The inflected stems point in the direction of 'deep' (etymological) bisyllabicity.

We proceed to the disyllabic nouns, first those with an open second syllable (Table 2), then those with a closed second syllable ('.' indicates the location of the syllable boundary).

The tendency to avoid long vowels and diphthongs in the second syllable of genuine underived noun roots is very strong. There are only a few handfuls of such words, fractions of one percent of 4,958. However, there are around 100 borrowings like *filee, revyy, turnee* and many bimorphemic derivatives like *takuu* 'guarantee' (from *takaa-*, inflectional stem, 'to guarantee').

The most striking fact of Table 2 is the prevalence of long (bimoraic) first syllables , i.e. the structures CVV.CV and especially CVC.CV which are much more frequent that than the theoretically simplest pattern CV.CV. The trimoraic pattern CVVC.CV is almost as frequent as monomoraic CV.CV, which must be considered very surprising. The share of monomoraic (C)V.CV is only 756 + 61 = 817, i.e. 16%. The four-moraic pattern (C)VVCC.CV is encountered only in a few borrowings. As is to be expected, V-initial first syllables are much more infrequent, by a factor of 10 – 20, than CV-initial first syllables, e.g. V.CV as compared to CV.CV. VV.V does not occur in underived words but the derivative *ai.e* 'intention' (*ai+e*, from *aiko-* 'intend')

Table 2. Disyllabic noun patterns in Finnish with an open second syllable.

Pattern	Number	Examples
CV.CV	756	kala, peto, maku
V.CV	61	aho, ele, äly
CVV.CV	950	jousi, laatu, määrä, tuoli
VV.CV	52	aamu, aika, ääni
VV.V	-	
CVC.CV	1795	hihna, kukko, pentu
VC.CV	143	ahma, olki, ämmä
CVVC.CV	628	haaska, juusto, lieska
VVC.CV	33	aalto, aitta, äänne
CVCC.CV	503	harppi, kalske, lamppu
VCC.CV	23	ankka, arkki, yrtti
(C)VVCC.CV	6	aortta, nyanssi, seanssi
X.CVV	7	ehtoo, harmaa, suklaa, vastuu, tienoo, Porvoo, vainaa
sum	4,958	

is a singleton example of this pattern. The number of underived bisyllabic nouns with a closed second syllable is around 800 of which some 650 have a bimoraic first syllable.

7.4 Conclusion

Finnish has only some 30 monosyllabic and less than 6,000 underived bisyllabic nouns. Somewhat surprisingly, we have demonstrated that the prototypical first syllable of mono- and disyllabic nouns is bimoraic rather than monomoraic. The latter would be expected on grounds relating to general phonological simplicity, i.e. the universal preference for optimal light CV-syllables. Trisyllabic and longer nouns have not been analyzed in detail here but a fast test shows that more than 75% of them too have bimoraic or even heavier first syllables. The same holds across the board for the vocabulary: 75% of the lexemes listed in RDF have at least a bimoraic first syllable (CVC. 40,378, CVV. 13,899, CV. 17,171).

Why are more complex phonotactic structures so clearly preferred over simpler ones? Three possible causes come to mind. First, languages with relatively few phonemes (e.g. Finnish with twenty-one) tend to have longer words than languages with more phonemes (Nettle 1999). Thus, in Nettle's sample of ten totally unrelated languages from different stocks, the mean word length of Khoisan !Kung with 147 phonemes in its inventory was 4.02 segments whereas that of Turkish with 28 phonemes was 6.44 segments, 'word' being defined as a random sample of fifty uninflected stems in a size-

able dictionary. Second, bimoraic (and longer) syllables amplify the effect of the word-stress fixed in Finnish to the first syllable. Third, for morphophonological reasons, new words and borrowings prefer quantitative over qualitative consonant gradation. Quantitative gradation is possible only in (at least) bimoraic syllables, e.g. *rokki* 'rock' - *roki+n* (genitive singular).

References

Karlsson, F. 1983. *Suomen kielen äänne- ja muotorakenne.* Porvoo Helsinki Juva: Werner Söderström osakeyhtiö.

Koskenniemi, K. 1978. Suomen kielen sananmuotojen perusmuodon automaattinen päättely. Mahdollisuuksien arviointia ja taivutuksen formalisointi. *Helsinki: Computing, Centre, University of Helsinki, Research Reports N:o 5.*

Koskenniemi, K. 1983. Two-level morphology: A General Computational Model for Word-Form Recognition and Production. *Helsinki: University of Helsinki, Department of General Linguistics, Publications No. 11.*

Nettle, D. 1999. *Linguistic Diversity.* Oxford: Oxford University Press.

Tuomi, T. 1972. Suomen kielen käänteissanakirja. Reverse Dictionary of Modern Standard Finnish. *Suomalaisen Kirjallisuuden Seuran Toimituksia. 274. osa.* Helsinki: Suomalaisen Kirjallisuuden Seura.

8

Twenty-Five Years of Finite-State Morphology

LAURI KARTTUNEN AND KENNETH R. BEESLEY

8.1 Introduction

Twenty-five years ago in the early 1980s, morphological analysis of natural language was a challenge to computational linguists. Simple cut-and-paste programs could be and were written to analyze strings in particular languages, but there was no general language-independent method available. Furthermore, cut-and-paste programs for analysis were not reversible, they could not be used to generate words. Generative phonologists of that time described morphological alternations by means of ordered rewrite rules, but it was not understood how such rules could be used for analysis.

This was the situation in the spring of 1981 when Kimmo Koskenniemi came to a conference on parsing that Lauri Karttunen had organized at the University of Texas at Austin. Also at the same conference were two Xerox researchers from Palo Alto, Ronald M. Kaplan and Martin Kay. The four Ks discovered that all of them were interested and had been working on the problem of morphological analysis. Koskenniemi went on to Palo Alto to visit Kay and Kaplan at Xerox PARC.

This was the beginning of Two-Level Morphology, the first general model in the history of computational linguistics for the analysis and generation of morphologically complex languages. The language-specific components, the lexicon and the rules, were combined with a runtime engine applicable to all languages.

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

8.2 The Origins

Traditional phonological grammars, formalized by Chomsky and Halle (1968), consisted of an ordered sequence of REWRITE RULES that converted abstract phonological representations into surface forms through a series of intermediate representations. Such rewrite rules have the general form $\alpha \rightarrow \beta \ / \ \gamma \ \_ \ \delta$ where α, β, γ, and δ can be arbitrarily complex strings or feature-matrices. The rule is read "α is rewritten as β between γ and δ". In mathematical linguistics (Partee et al. 1993), such rules are called CONTEXT-SENSITIVE REWRITE RULES, and they are more powerful than regular expressions or context-free rewrite rules.

In 1972, C. Douglas Johnson published his dissertation, *Formal Aspects of Phonological Description*, wherein he showed that phonological rewrite rules are actually much less powerful than the notation suggests. Johnson observed that while the same context-sensitive rule could be applied several times recursively to its own output, phonologists have always assumed implicitly that the site of application moves to the right or to the left in the string after each application. For example, if the rule $\alpha \rightarrow \beta \ / \ \gamma \ \_ \ \delta$ is used to rewrite the string $\gamma\alpha\delta$ as $\gamma\beta\delta$, any subsequent application of the same rule must leave the β part unchanged, affecting only γ or δ. Johnson demonstrated that the effect of this constraint is that the pairs of inputs and outputs produced by a phonological rewrite rule can be modeled by a finite-state transducer. This result was largely overlooked at the time and was rediscovered by Ronald M. Kaplan and Martin Kay around 1980 . Putting things into a more algebraic perspective than Johnson, Kaplan and Kay showed that phonological rewrite rules describe REGULAR RELATIONS. By definition, a regular relation can be represented by a finite-state transducer.

Johnson was already aware of an important mathematical property of finite-state transducers established by Schützenberger (1961): there exists, for any pair of transducers applied sequentially, an equivalent single transducer. Any cascade of rule transducers can in principle be composed into a single transducer that maps lexical forms directly into the corresponding surface forms, and vice versa, without any intermediate representations.

These theoretical insights did not immediately lead to practical results. The development of a compiler for rewrite rules turned out to be a very complex task. It became clear that building a compiler required as a first step a complete implementation of basic finite-state operations such as union, intersection, complementation, and composition. Developing a complete finite-state calculus was a challenge in itself on the computers that were available at the time.

Another reason for the slow progress may have been that there were persistent doubts about the practicality of the approach for morphological ANAL-

YSIS. Traditional phonological rewrite rules describe the correspondence between lexical forms and surface forms as a one-directional, sequential mapping from lexical forms to surface forms. Even if it was possible to model the GENERATION of surface forms efficiently by means of finite-state transducers, it was not evident that it would lead to an efficient analysis procedure going in the reverse direction, from surface forms to lexical forms.

Let us consider a simple illustration of the problem with two sequentially applied rewrite rules, `N -> m / _ p` and `p -> m / m _`. The corresponding transducers map the lexical form *kaNpat* unambiguously to *kammat*, with *kampat* as the intermediate representation. However if we apply the same transducers in the other direction to the input *kammat*, we get the three results shown in Figure 1.

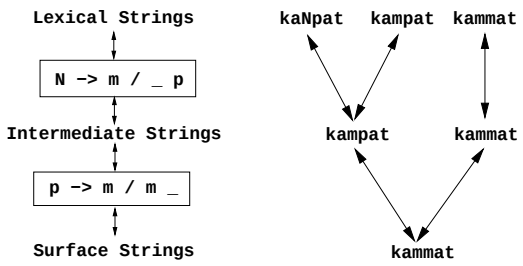


FIGURE 1 Deterministic Generation, Nondeterministic Analysis

This asymmetry is an inherent property of the generative approach to phonological description. If all the rules are deterministic and obligatory and if the order of the rules is fixed, each lexical form generates only one surface form. But a surface form can typically be generated in more than one way, and the number of possible analyses grows with the number of rules that are involved. Some of the analyses may turn out to be invalid because the putative lexical forms, say *kammat* and *kampat* in this case, might not exist in the language. But in order to look them up in the lexicon, the system must first complete the analysis. Depending on the number of rules involved, a surface form could easily have dozens of potential lexical forms, even an infinite number in the case of certain deletion rules.

Although the generation problem had been solved by Johnson, Kaplan and Kay, at least in principle, the problem of efficient morphological analysis in the Chomsky-Halle paradigm was still seen as a formidable challenge. As counterintuitive as it was, it appeared that analysis was computationally a much more difficult task than generation. Composing all the rule transducers into a single one would not solve the “overanalysis” problem. Because the resulting single transducer is equivalent to the original cascade, the ambiguity

remains.

The solution to the overanalysis problem should have been obvious: to formalize the lexicon itself as a finite state transducer and compose the lexicon with the rules. In this way, all the spurious ambiguities produced by the rules are eliminated at compile time. The resulting single transducer contains only lexical forms that actually exist in the language. When this idea first surfaced in Karttunen et al. (1992), it was not in connection with traditional rewrite rules but with an entirely different finite-state formalism that had been introduced in the meantime, called TWO-LEVEL RULES (Koskenniemi 1983).

8.3 Two-level Morphology

In the spring of 1981 when Kimmo Koskenniemi came to the USA for a visit, he learned about Kaplan and Kay's finite-state discovery.[1] PARC had begun work on the finite-state algorithms, but they would prove to be many years in the making. Koskenniemi was not convinced that efficient morphological analysis would ever be practical with generative rules, even if they were compiled into finite-state transducers. Some other way to use finite automata might be more efficient.

Back in Finland, Koskenniemi invented a new way to describe phonological alternations in finite-state terms. Instead of cascaded rules with intermediate stages and the computational problems they seemed to lead to, rules could be thought of as statements that directly constrain the surface realization of lexical strings. The rules would not be applied sequentially but in parallel. Each rule would constrain a certain lexical/surface correspondence and the environment in which the correspondence was allowed, required, or prohibited. For his 1983 dissertation, Koskenniemi constructed an ingenious implementation of his constraint-based model that did not depend on a rule compiler, composition or any other finite-state algorithm, and he called it TWO-LEVEL MORPHOLOGY. Two-level morphology is based on three ideas:

- Rules are symbol-to-symbol constraints that are applied in parallel, not sequentially like rewrite rules.
- The constraints can refer to the lexical context, to the surface context, or to both contexts at the same time.
- Lexical lookup and morphological analysis are performed in tandem.

To illustrate the first two principles we can turn back to the *kaNpat* example again. A two-level description of the lexical-surface relation is sketched in Figure 2.

As the lines indicate, each symbol in the lexical string *kaNpat* is paired with its realization in the surface string *kammat*. Two of the symbol pairs in Fig-

[1] They weren't then aware of Johnson's 1972 publication.

FIGURE 2 Example of Two-Level Constraints

ure 2 are constrained by the context marked by the associated box. The *N:m* pair is *restricted* to the environment having an immediately following *p* on the lexical side. In fact the constraint is tighter. In this context, all other possible realizations of a lexical *N* are *prohibited*. Similarly, the *p:m* pair requires the preceding surface *m*, and no other realization of *p* is allowed here. The two constraints are independent of each other. Acting in parallel, they have the same effect as the cascade of the two rewrite rules in Figure 1. In Koskenniemi's notation, these rules are written as `N:m <=> _ p:` and `p:m <=> :m _` , where `<=>` is an operator that combines a context restriction with the prohibition of any other realization for the lexical symbol of the pair. The colon in the right context of first rule, `p:`, indicates that it refers to a lexical symbol; the colon in the left context of the second rule, `:m`, indicates a surface symbol.

Two-level rules may refer to both sides of the context at the same time. The *y*∼*ie* alternation in English plural nouns could be described by two rules: one realizes *y* as *i* in front of an epenthetic *e*; the other inserts an epenthetic *e* between a lexical consonant-*y* sequence and a morpheme boundary (+) that is followed by an *s*. Figure 3 illustrates the *y:i* and *0:e* constraints.

FIGURE 3 A Two-Level View of *y*∼*ie* Alternation in English

Note that the *e* in Figure 3 is paired with a *0* (= zero) on the lexical level. In two-level rules, zero is a symbol like any other; it can be used to constrain the realization of other symbols, as in `y:i <=> _ 0:e`. In fact, all the other rules must "know" where zeros may occur. Zeros are treated as epsilons only when two-level rules are applied to strings.

Like rewrite rules, two-level rules describe regular relations; but there is an important difference. Because the zeros in two-level rules are ordinary symbols, a two-level rule represents an EQUAL-LENGTH RELATION. This has an important consequence: Although regular relations in general are not closed under intersection, equal length relations have that property. When a set of

two-level transducers are applied in parallel, the apply routine in fact simulates the intersection of the rule automata and composes the input string with the virtual constraint network.

Applying the rules in parallel does not in itself solve the overanalysis problem discussed in the previous section. The two constraints sketched above allow *kammat* to be analyzed as *kaNpat*, *kampat*, or *kammat*. However, the problem becomes manageable when there are no intermediate levels of analysis. In Koskenniemi's 1983 system, the lexicon was represented as a forest of tries (= letter trees), tied together by continuation-class links from leaves of one tree to roots of another tree or trees.[2] Lexical lookup and the analysis of the surface form are performed in tandem. In order to arrive at the point shown in Figure 4, the analyzer has traversed a branch in the lexicon that

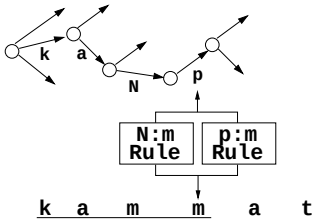


FIGURE 4 Following a Path in the Lexicon

contains the lexical string *kaN*. At this point, it only considers symbol pairs whose lexical side matches one of the outgoing arcs of the current state. It does not pursue analyses that have no matching lexical path.

Koskenniemi's two-level morphology was the first practical general model in the history of computational linguistics for the analysis of morphologically complex languages. The language-specific components, the rules and the lexicon, were combined with a universal runtime engine applicable to all languages.

8.4 A Two-Level Rule Compiler

In his dissertation, Koskenniemi introduced a formalism for two-level rules. The semantics of two-level rules was well-defined but there was no rule compiler available at the time. Koskenniemi and other early practitioners of two-level morphology constructed their rule automata *by hand*. This is tedious in the extreme and very difficult for all but very simple rules.

Although two-level rules are formally quite different from the rewrite rules studied by Kaplan and Kay, the methods that had been developed for com-

[2]The TEXFIN analyzer developed at the University of Texas at Austin (Karttunen et al. 1981) had the same lexicon architecture.

piling rewrite rules were applicable to two-level rules as well. In both formalisms, the most difficult case is a rule where the symbol that is replaced or constrained appears also in the context part of the rule. This problem Kaplan and Kay had already solved by an ingenious technique for introducing and then eliminating auxiliary symbols to mark context boundaries. Another fundamental insight they had was the encoding of context restrictions in terms of double negation. For example, a constraint such as "*p* must be followed by *q*" can be expressed as "it is not the case that something ending in *p* is not followed by something starting with *q*." In Koskenniemi's formalism, `p => _ q`.

In the summer of 1985, when Koskenniemi was a visitor at Stanford, Kaplan and Koskenniemi worked out the basic compilation algorithm for two-level rules. The first two-level rule compiler was written in InterLisp by Koskenniemi and Karttunen in 1985-87 using Kaplan's implementation of the finite-state calculus (Koskenniemi 1986, Karttunen et al. 1987). The current C-version of the compiler, called TWOLC, was written at PARC in 1991-92 (Karttunen and Beesley 1992).[3]

Although the basic compilation problem was solved quickly, building a practical compiler for two-level rules took a long time. The TWOLC compiler includes sophisticated techniques for checking and resolving conflicts between rules whenever possible. Without these features, large rule systems would have been impossible to construct and debug. If two constraints are in conflict, some lexical forms have no valid surface form. This is a common problem and often difficult to remedy even if the compiler is able to detect the situation and to pinpoint the cause.

8.5 Two-Level Implementations

Koskenniemi's Pascal implementation was quickly followed by others. The most influential of them was the KIMMO system by Lauri Karttunen and his students at the University of Texas (Karttunen 1983, Gajek et al. 1983). This Lisp project inspired many copies and variations, including those by Beesley (1989, 1990). A free C implementation of classic Two-Level Morphology, called PC-KIMMO, from the Summer Institute of Linguistics (Antworth 1990), became a popular tool.

In Europe, two-level morphological analyzers became a standard com-

[3]The landmark 1994 article by Kaplan and Kay on the mathematical foundations of finite-state linguistics defines the basic compilation algorithm for phonological rewrite rules and for Koskenniemi's two-level rules. The article appeared years after the work on the two-level compiler was completed and just before the implementation of the so-called REPLACE RULES in the current PARC/XRCE regular expression compiler. The article is accurate on the former topic, but the algorithm for replace rules (Karttunen 1995, 1996, Kempe and Karttunen 1996) differs in many details from the compilation of rewrite rules as described by Kaplan and Kay.

ponent in several large systems for natural language processing such as the British Alvey project (Black et al. 1987, Ritchie et al. 1987, 1992), SRI's CLE Core Language Engine (Carter 1995), the ALEP Natural Language Engineering Platform (Pulman 1991) and the MULTEXT project (Armstrong 1996). ALEP and MULTEXT were funded by the European Commission. [4]

Some of these systems were based on simplified two-level rules, the so-called PARTITION-BASED formalism Ruessink (1989), which was claimed to be easier for linguists to learn than the original Koskenniemi notation. But none of these systems had a finite-state rule compiler.[5] Another difference was that morphological parsing could be constrained by feature unification. Because the rules were interpreted at runtime and because of the unification overhead, these systems were not efficient, and two-level morphology acquired, undeservedly, a reputation for being slow.

At XRCE and Inxight, the TWOLC compiler was used in the 1990s to develop comprehensive morphological analyzer for numerous languages. Another utility, called LEXC (Karttunen 1993b), made it possible to combine a finite-state lexicon with a set of two-level rules into a single LEXICAL TRANSDUCER using a special "intersecting composition" algorithm that simulates the intersection of the rules while simultaneously composing the virtual rule transducer with the lexicon. A lexical transducer can be considered the ultimate two-level model of a language as it encodes compactly:

- all the LEMMAS (canonical lexical forms with morphological tags)
- all the inflected surface forms
- all the mappings between lexical forms and surface forms.

In the course of this work it became evident that lexical transducers are easier to construct with sequentially applied replace rules than with two-level rules. Large systems of two-level rules are notoriously difficult to debug. Most developers of morphological analyzers at XRCE and at companies such as Inxight have over the years switched to the sequential model and the XFST tool that includes a compiler for replace rules. The ordering of replace rules seems to be less of a problem than the mental discipline required to avoid rule conflicts in a two-level system, even if the compiler automatically resolves most of them. From a formal point of view there is no substantive difference; a cascade of rewrite rules and a set of parallel two-level constraints are just two different ways to decompose a complex regular relation into a set of simpler relations that are easier to understand and manipulate.

[4]The MULTEXT morphology tool (Petitpierre and Russel 1995) built at ISSCO is available at `http://packages.debian.org/stable/misc/mmorph.html`

[5]A compilation algorithm has been developed for the partition-based formalism Grimley-Evans et al. (1996), but to our knowledge there is no publicly available implementation.

The Beesley and Karttunen (2003) book *Finite State Morphology* describes the XFST and LEXC tools and offers a lot of practical advice on techniques for constructing lexical transducers.[6]

8.6 Reflections

Although the two-level approach to morphological analysis was quickly accepted as a useful practical method, the linguistic insight behind it was not picked up by mainstream linguists. The idea of rules as parallel constraints between a lexical symbol and its surface counterpart was not taken seriously at the time outside the circle of computational linguists. Many arguments had been advanced in the literature to show that phonological alternations could not be described or explained adequately without sequential rewrite rules. It went largely unnoticed that two-level rules could have the same effect as ordered rewrite rules because two-level rules allow the realization of a lexical symbol to be constrained either by the lexical side or by the surface side. The standard arguments for rule ordering were based on the *a priori* assumption that a rule could refer only to the input context (Karttunen 1993a).

But the world has changed. Current phonologists, writing in the framework of OT (Optimality Theory), are sharply critical of the "serialist" tradition of ordered rewrite rules that Johnson, Kaplan and Kay wanted to formalize (Prince and Smolensky 1993, Kager 1999, McCarthy 2002).[7] In a nutshell, OT is a two-level theory with *ranked* parallel constraints. Many types of optimality constraints can be represented trivially as two-level rules. In contrast to Koskenniemi's "hard" constraints, optimality constraints are "soft" and violable. There are of course many other differences. Most importantly, OT constraints are meant to be universal. The fact that two-level rules can describe orthographic idiosyncrasies such as the *y*∼*ie* alternation in English with no appeal to universal principles is a minus rather than a plus. It makes the approach uninteresting from the OT point of view.[8]

Nevertheless, from the OT perspective, two-level rules have some interesting properties. They are symbol-to-symbol constraints, not string-to-string relations like general rewrite rules. Two-level rules enable the linguist to refer to the input and the output context in the same constraint. The notion of FAITHFULNESS (= no change) can be expressed straight-forwardly. It is possible to formulate constraints that constrain directly the surface level. These ideas were ten years ahead of their time in 1983.

[6]The book includes a CD that contains TWOLC, XFST, LEXC and other finite-state tools. See also `http://www.fsmbook.com`. The documentation for TWOLC, missing from the book, is included on the CD.

[7]The term SERIAL, a pejorative term in an OT context, refers to sequential rule application.

[8]Finite-state approaches to Optimality Theory have been explored in several recent articles (Eisner 1997, Frank and Satta 1998, Karttunen 1998).

It is interesting to observe that computational linguists and “paper-and-pencil linguists” have historically been out of sync in their approach to phonology and morphology. When computational linguists implemented parallel two-level models in the 1980s, paper-and-pencil linguists were still stuck in the serialist Chomsky-Halle paradigm. When most of the computational morphologists working with the Xerox tools embraced the sequential model as the more practical approach in the mid 1990s, a two-level theory took over paper-and-pencil linguistics by a storm in the guise of OT.

If one views the mapping from lexical forms to surface forms as a regular relation, the choice between different ways of decomposing it has practical consequences but it is not a deep theoretical issue for computational linguists. No brand of finite-state morphology has ever been promoted as a theory about language. Its practitioners have always been focused on the practical task of representing the morphological aspects of a language in a form that supports efficient analysis and generation. They have been remarkably successful in that task.

Paper-and-pencil morphologists in general are not interested in creating complete descriptions for particular languages. They design formalisms for expressing generalizations about morphological phenomena commonly found in all natural languages. But if it turns out, as in the case of REALIZATIONAL MORPHOLOGY (Stump 2001), that the theory can be implemented with finite-state tools (Karttunen 2003), perhaps the phenomena are not as complex as the linguist has imagined.

Acknowledgments

Much of the material in this article comes from an unpublished paper, "A Short History of Two-Level Morphology", that we presented at a Special Event celebrating "Twenty Years of Two-Level Morphology" at ESSLLI-2001 in Helsinki (`http://www.helsinki.fi/esslli/`).

References

Antworth, Evan L. 1990. *PC-KIMMO: a two-level processor for morphological analysis*. No. 16 in Occasional publications in academic computing. Dallas: Summer Institute of Linguistics.

Armstrong, Susan. 1996. Multext: Multilingual text tools and corpora. In H. Feldweg and E. W. Hinrichs, eds., *Lexikon und Text*, pages 107–112. Tuebingen: Max Niemeyer Verlag.

Beesley, Kenneth R. 1989. Computer analysis of Arabic morphology: A two-level approach with detours. In *Third Annual Symposium on Arabic Linguistics*. Salt Lake City: University of Utah. Published as Beesley, 1991.

Beesley, Kenneth R. 1990. Finite-state description of Arabic morphology. In *Proceedings of the Second Cambridge Conference on Bilingual Computing in Arabic and English*. No pagination.

Beesley, Kenneth R. 1991. Computer analysis of Arabic morphology: A two-level approach with detours. In B. Comrie and M. Eid, eds., *Perspectives on Arabic Linguistics III: Papers from the Third Annual Symposium on Arabic Linguistics*, pages 155–172. Amsterdam: John Benjamins.

Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite State Morphology*. Palo Alto, CA: CSLI Publications.

Black, A., G. Ritchie, S. Pulman, and G. Russell. 1987. Formalisms for morphographemic description. In *Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics*, pages 11–18.

Carter, D. 1995. Rapid development of morphological descriptions for full language processing systems. In *Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics*, pages 202–209.

Chomsky, Noam and Morris Halle. 1968. *The Sound Pattern of English*. New York: Harper and Row.

Eisner, Jason. 1997. Efficient generation in primitive Optimality Theory. In *Proceedings of the 35th Annual ACL and 8th EACL*, pages 313–320. Madrid: Association for Computational Linguistics.

Frank, Robert and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics* 24(2):307–316.

Gajek, Oliver, Hanno T. Beck, Diane Elder, and Greg Whittemore. 1983. LISP implementation. *Texas Linguistic Forum* 22:187–202.

Grimley-Evans, Edmund, George Anton Kiraz, and Steven G. Pulman. 1996. Compiling a partition-based two-level formalism. In *COLING'96*. Copenhagen. cmp-lg/9605001.

Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description*. The Hague: Mouton.

Kager, Reni. 1999. *Optimality Theory*. Cambridge, England: Cambridge University Press.

Kaplan, Ronald M. and Martin Kay. 1981. Phonological rules and finite-state transducers. In *Linguistic Society of America Meeting Handbook, Fifty-Sixth Annual Meeting*. New York. Abstract.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20(3):331–378.

Karttunen, Lauri. 1983. KIMMO: a general morphological processor. *Texas Linguistic Forum* 22:165–186.

Karttunen, Lauri. 1993a. Finite-state constraints. In J. Goldsmith, ed., *The Last Phonological Rule*. Chicago, Illinois.: University of Chicago Press.

Karttunen, Lauri. 1993b. Finite-state lexicon compiler. Tech. Rep. ISTL-NLTT-1993-04-02, Xerox Palo Alto Research Center, Palo Alto, CA.

Karttunen, Lauri. 1995. The replace operator. In *ACL'95*. Cambridge, MA. cmp-lg/9504032.

Karttunen, Lauri. 1996. Directed replacement. In *ACL'96*. Santa Cruz, CA. cmp-lg/9606029.

Karttunen, Lauri. 1998. The proper treatment of optimality in computational phonology. In *FSMNLP'98. International Workshop on Finite-State Methods in Natural Language Processing*. Bilkent University, Ankara, Turkey. cmp-lg/9804002.

Karttunen, Lauri. 2003. Computing with realizational morphology. In A. Gelbukh, ed., *Computational Linguistics and Intelligent Text Processing*, vol. 2588 of *Lecture Notes in Computer Science*, pages 205–216. Heidelberg: Springer Verlag.

Karttunen, Lauri and Kenneth R. Beesley. 1992. Two-level rule compiler. Tech. Rep. ISTL-92-2, Xerox Palo Alto Research Center, Palo Alto, CA.

Karttunen, Lauri, Ronald M. Kaplan, and Annie Zaenen. 1992. Two-level morphology with composition. In *COLING'92*, pages 141–148. Nantes, France.

Karttunen, Lauri, Kimmo Koskenniemi, and Ronald M. Kaplan. 1987. A compiler for two-level phonological rules. In M. Dalrymple, R. Kaplan, L. Karttunen, K. Koskenniemi, S. Shaio, and M. Wescoat, eds., *Tools for Morphological Analysis*, vol. 87-108 of *CSLI Reports*, pages 1–61. Palo Alto, CA: Center for the Study of Language and Information, Stanford University.

Karttunen, Lauri, Hans Uszkoreit, and Rebecca Root. 1981. Morphological analysis of Finnish by computer. In *Proceedings of the 71st Annual Meeting of SASS*. Albuquerque, New Mexico.

Kempe, André and Lauri Karttunen. 1996. Parallel replacement in finite-state calculus. In *COLING'96*. Copenhagen. cmp-lg/9607007.

Koskenniemi, Kimmo. 1983. Two-level morphology: A general computational model for word-form recognition and production. Publication 11, University of Helsinki, Department of General Linguistics, Helsinki.

Koskenniemi, Kimmo. 1986. Compilation of automata from morphological two-level rules. In F. Karlsson, ed., *Papers from the Fifth Scandinavian Conference on Computational Linguistics*, pages 143–149.

McCarthy, John J. 2002. *The Foundations of Optimality Theory*. Cambridge, England: Cambridge University Press.

Partee, B. H., A. ter Meulen, and R. E. Wall. 1993. *Mathematical Methods in Linguistics*. Dordrecht: Kluwer.

Petitpierre, Dominique and Graham Russel. 1995. MMORPH — the multext morphology program. Multext Deliverable 2.3.1, ISSCO, Carouge, Switzerland.

Prince, Allan and Paul Smolensky. 1993. Optimality Theory: Constraint interaction in generative grammar. Tech. rep., Rutgers University, Piscataway, NJ. RuCCS Technical Report 2. Rutgers Center for Cognitive Science.

Pulman, S. 1991. Two level morphology. In H. Alshawi, D. Arnold, R. Backofen, D. Carter, J. Lindop, K. Netter, S. Pulman, J. Tsujii, and H. Uskoreit, eds., *ET6/1 Rule Formalism and Virtual Machine Design Study*, chap. 5. Luxembourg: CEC.

Ritchie, G., A. Black, S. Pulman, and G. Russell. 1987. The Edinburgh/Cambridge morphological analyser and dictionary system (version 3.0) user manual. Tech. Rep. Software Paper No. 10, Department of Artificial Intelligence, University of Edinburgh.

Ritchie, G., G. Russell, A. Black, and S. Pulman. 1992. *Computational Morphology: Practical Mechanisms for the English Lexicon*. MIT Press, Cambridge, Mass.

Ruessink, H. 1989. Two level formalisms. Utrecht Working Papers in NLP 5, Utrecht University.

Schützenberger, Marcel-Paul. 1961. A remark on finite transducers. *Information and Control* 4:185–196.

Stump, Gregory T. 2001. *Inflectional Morphology. A Theory of Paradigm Structure*. Cambridge, England: Cambridge University Press.

9

Local Grammar Algorithms

MEHRYAR MOHRI

Local syntactic constraints can often be described with much precision. For example, the sequences of preverbal particles in French, e.g., *ne, le, lui,* obey strict rules that govern their ordering and the insertion of other terms among them, regardless of the remaining material forming the sentence (Gross, 1968). As such, they can be viewed as *local rules*, or *local grammars*. Similar detailed local grammars have been given for time expressions (Maurel, 1989) and later for many other linguistic expressions (Gross, 1997).

Such *local grammars* do not just capture some rare linguistic phenomena. Widespread technical jargon, idioms, or clichés, lead to common syntactic constraints that can be accurately described locally without resorting to more powerful syntactic formalisms. A careful examination of articles written in the financial section of a newspaper reveals for example that only a limited number of constructions accounts for the description of the variations of the stock market, or the changes in inflation or unemployment rate.

A local grammar may describe a set of forbidden or unavoidable sequences. In both cases, it can be compactly represented by a finite automaton. A collection of local grammars can be combined and represented by a more complex finite automaton by taking the union of the simpler local grammar automata. Novel linguistic studies keep increasing the number of local grammars (Gross, 1997). This tends to significantly increase the size of the union local grammar finite automata and creates the need for efficient algorithms to apply large local grammar automata.

This chapter presents an overview of algorithms for the application of local grammar automata. Section 1 introduces the algorithmic problem related to the application of large local grammar automata. Section 2 reviews two

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

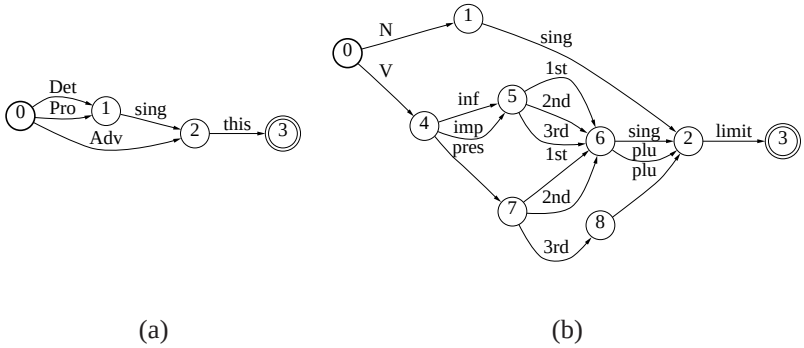


FIGURE 1 Automata representing the possible part-of-speech tags for (a) *this*; and (b) *limit*; in the absence of any context.

local grammar algorithms and presents in detail the most efficient one. It also illustrates these algorithms by showing examples of their applications.

9.1 Description of the problem

Let Σ denote the alphabet and let A be a local grammar finite automaton specifying a set of forbidden sequences. We denote by $L(A)$ the language accepted by A. By definition, any sequence containing a sequence accepted by A is unacceptable. Thus, acceptable sequences must be in $\overline{\Sigma^* L(A) \Sigma^*}$.

Let us illustrate this with an example related to part-of-speech tagging. Let T be the automaton representing the set of all possible tagging of a text (Koskenniemi, 1990). T can be obtained by concatenating simpler automata representing the set of possible tagging for each of the word composing the text. Figures 1(a)-(b) show these automata for the words *this* and *limit*. The three paths of the automaton of Figure 1(a) account for the fact that *this* may be a singular (sing) determiner (Det) or pronoun (Pro), or an adverb (Adv) as in: *Tom is this tall*. Similarly, the automaton of Figure 1(b) has different paths corresponding to the case where the word *limit* is a singular (sing) noun (N), or the infinitive (inf), imperative (imp), or present (pres) form of a verb (V), with the third (3rd) person singular (sing) form excluded in the latter case.

Simple observations can help derive a set of forbidden sequences represented by the automaton A of Figure 2. For example, when *this* is an adverb, it cannot be followed by a noun or a verb, and similarly, when it is a determiner, it cannot precede a verb unless the verb is a past or present participle. The automaton A can help reduce the ambiguities of the text T since it enforces that the sequences accepted must be in $L(T) \cap \overline{\Sigma^* L(A) \Sigma^*}$. Figure 3 shows the automaton of accepted sequences resulting from the application of the local grammar A to T.

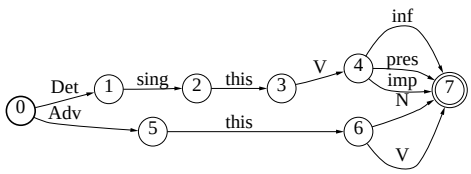


FIGURE 2 Finite automaton A representing a set of forbidden sequences.

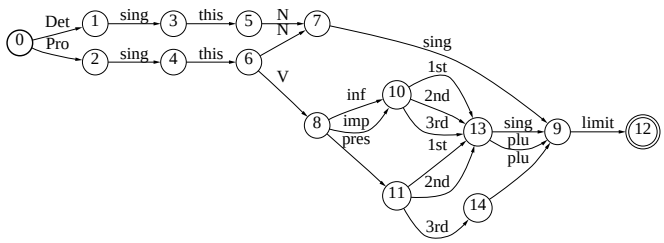


FIGURE 3 Accepted sequences $L(T) \cap \overline{\Sigma^* L(A) \Sigma^*}$.

The main problem for the application of a large local grammar A to a text automaton T is the efficient computation of an automaton representing $L(T) \cap \overline{\Sigma^* L(A) \Sigma^*}$. Complex local grammars automata may have in the order of several million transitions. The alphabet includes the vocabulary of the language considered, which, in English, has more than 200,000 words.

An automaton accepting $\Sigma^* L(A) \Sigma^*$ can thus be very large. Taking the complement of that automaton may lead to an even larger automaton since the worst case complexity of complementation is exponential for a non-deterministic automaton, and the result would yet need to be intersected with T.

The next section examines several algorithms for the computation of an automaton representing $L(T) \cap \overline{\Sigma^* L(A) \Sigma^*}$.

9.2 Algorithms

This section presents two local grammar algorithms. It first discusses the properties of a simple algorithm that can be viewed as the counter-part for local grammar algorithms of the straightforward quadratic-time string-matching algorithm and illustrates its application. A more efficient algorithm is then described in detail, including its pseudocode, and its optimization. In

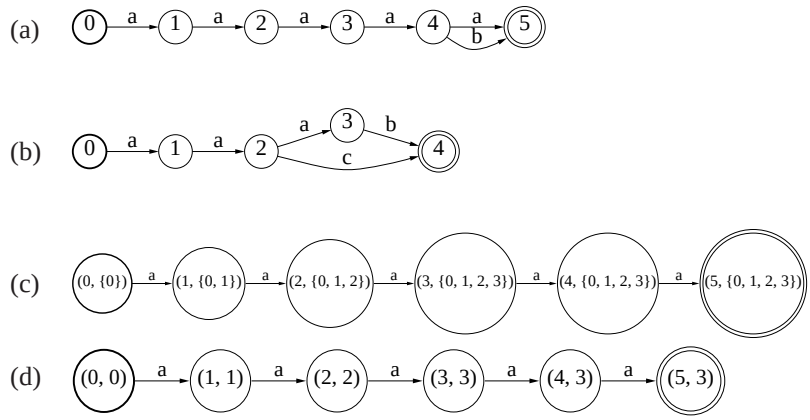


FIGURE 4 (a) Simple text automaton T_0. (b) Simple local grammar automaton A_0. (c) Result of the application of A_0 to T_0 using the simple algorithm. (d) Application of A_0 to T_0 using a more efficient algorithm.

what follows, the local grammar automaton A and the text automaton T will be assumed to be deterministic.

9.2.1 A simple algorithm

The problem of the application of a local grammar can be viewed as a generalization to automata of pattern-matching in text. A simple algorithm for the application of A to T is to search for all sequences accepted by A starting from each state of T. If a forbidden sequence is found, the appropriate transition is removed to disallow that sequence. This can be done by:

- simulating the presence of a self-loop labeled with all elements of Σ at the initial state of A;
- reading the paths of T starting from its initial state while pairing each state reached by a string x with the set of all states of A that can be reached by x from its initial state.

This describes the algorithm of Roche (1992). Figure 4(c) shows its result when using the simple text automaton of Figure 4(a) and the local grammar A_0 shown in Figure 4(b). Each state of the output automaton is a pair (p, s) where p is a state of T and s an element of the powerset of the states of A. At each state, the transitions of state p and those of the set of states in s are matched to form new transitions. In general, this operation may be very costly because of the large number of transitions leaving the states of s. Note that the transition labeled with b from the state $(4, \{0, 1, 2, 3\})$ to $(5, \{4\})$ is not constructed to disallow the forbidden sequence $aaab$ (state 4 is a final state of

A_0).

As it is clear from this example, the algorithm is very similar to the simple quadratic-time string-matching algorithm seeking to match a pattern at each position of the text, ignoring the information derived from matching attempts at previous positions.

The next section describes an algorithm that precisely exploits such information as with the linear-time string-matching algorithm of Knuth et al. (1977). Figure 4(d) shows the result of the application of that algorithm. Each state of the output automaton is identified with a pair of states (p, q) where p is a state of T and q the state of A corresponding to the longest (proper) suffix of the strings leading to p.

9.2.2 A more efficient local grammar algorithm

The application of a local grammar is directly related to the computation of a deterministic automaton representing $\Sigma^* L(A)$. Let A' be the automaton constructed by augmenting A with a self-loop at its initial state labeled with all elements of the alphabet Σ, and let $B = det(A')$ be the result of the determinization of A'. B recognizes the language $\Sigma^* L(A)$. To apply the local grammar A to T, we can proceed as for computing the intersection $B \cap T$, barring the creation of transitions leading to a state identified with a pair (p, q) where q is a final state of B.

In fact, since determinization can be computed on-the-fly (Aho et al., 1986, Mohri, 1997a), the full determinization of A' is not needed, only the part relevant to the computation of the intersection with T. However, if one wishes to apply the grammar to many different texts, it is preferable to compute B' once beforehand. In general, the computation of B' may be very costly though, in particular because of the alphabet size $|\Sigma|$ which can reach several hundred thousand.

There exists an algorithm for constructing a compact representation of the deterministic automaton representing $\Sigma^* L(A)$ using failure transitions (Mohri, 1997b). A failure transition is a special transition that is taken when no standard transition with the desired input label is found.

The algorithm can be viewed as a generalization to an arbitrary deterministic automaton A of the classical algorithms of Knuth et al. (1977) and that of Aho and Corasick (1975) that were designed only for strings or trees. When A is a tree, the complexity of the algorithm of Mohri (1997b) coincides with that of Aho and Corasick (1975): it is linear in the sum of the lengths of the strings accepted by A.

The following is the pseudocode of that algorithm in the case where A is acyclic.

LocalGrammar(A)

```
1  E ← E ∪ {(i, φ, i)}
2  ENQUEUE(S, i)
3  while S ≠ ∅ do
4          p ← DEQUEUE(S)
5          for e ∈ E[p] do
6              q ← δ(p, φ)
7              while q ≠ i and δ(q, l[e]) = UNDEFINED do q ← δ(p, φ)
8              if p ≠ i and δ(q, l[e]) ≠ UNDEFINED
9                 then q ← δ(q, l[e])
10             if δ(n[e], φ) = UNDEFINED
11                then δ(n[e], φ) ← q
12                       if q ∈ F then F ← F ∪ {n[e]}
13                       L[n[e]] = L[n[e]] ∪ {n[e]}
14                       ENQUEUE(S, n[e])
15             else if there exists r ∈ L[old[n[e]]] such that (r, φ, q) ∈ E
16                  then n[e] ← r
17             else if old[q] ≠ n[e]
18                  then create new state r
19                           for e′ ∈ E[n[e]] such that l[e′] ≠ φ do
20                               E ← E ∪ {(r, l[e′], old[n[e′]])}
21                           E ← E ∪ (r, φ, q)
22                           old[r] ← old[n[e]]
23                           if old[n[e]] ∈ F then F ← F ∪ {r}
24                           L[old[n[e]]] = L[old[n[e]]] ∪ {r}
25                           n[e] ← r
26                           ENQUEUE(S, r)
27             else n[e] ← q
```

The algorithm takes as input a deterministic automaton A that it modifies to construct the desired local grammar automaton. States of A are visited in the order of a breadth-first search using a FIFO queue S. Each state q admits a failure transition labeled with ϕ. The destination state of that transition is the *failure state* of q, which is defined as the state reached by the longest proper suffix of the strings reaching q that are prefix of $L(A)$. Two distinct paths reaching q may correspond to two distinct failure states for q. In that case, q must be duplicated. Thus, the algorithm maintains the two following attributes: $old[q]$, the original state from which q was copied and, if q was originally in A (*i.e.* $old[q] = q$), $L[q]$, the list of the states obtained by copying q.

The outgoing transitions e of each state p extracted from the queue S (line 4) are examined. The candidate failure state q of $n[e]$ is determined (lines 6-

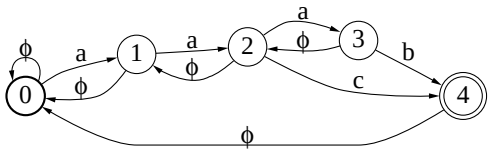


FIGURE 5 Finite automaton B'_0 recognizing $\Sigma^* L(A_0)$, where A_0 is the automaton of Figure 4. Failure transitions are marked with ϕ.

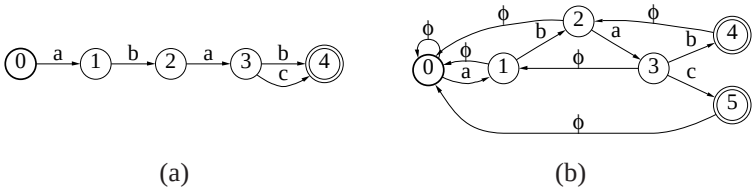


FIGURE 6 (a) Deterministic automaton A; (b) deterministic automaton B recognizing $\Sigma^* L(A)$. Transitions labeled with ϕ represent failure transitions.

10) as the first state on the failure path of p that has an outgoing transition labeled by $l[e]$. If $n[e]$ is not already assigned a failure state, its failure state is set to q and $n[e]$ is added to the queue (lines 10-14). If there exists a state r that has the same original state as $n[e]$ and has q as a failure state, then the destination of e is changed to r (lines 15-16). If q is not a copy of $n[e]$, then a new state r is created by copying $n[e]$, the failure state of r is set to q, the destination state of e is changed to r and r is added to the queue (lines 17-26). Otherwise, the destination state of e is changed to q (line 27).

When A is not acyclic, the condition of the test of line 17 needs to be generalized as described in detail in (Mohri, 1997b). An efficient implementation of this algorithm has been incorporated in the GRM library (Allauzen et al., 2005) with the command-line utility `grmlocalgrammar`.

Figure 5 shows the output of the algorithm when applied to the automaton A_0 of Figure 4(b). Each state admits a failure transition. The failure transition at the initial state is a self-loop. In such cases, the search for a default state can stop, e.g., at state 0, if a desired label such as b cannot be found, no further default state is considered. The automaton of Figure 5 is intersected with T_0 in the way previously described to produce the result (Figure 4(d)).

Figure 6(b) shows another illustration of the application where it is applied to the automaton of Figure 6(a). The special symbol ϕ is used to mark failure

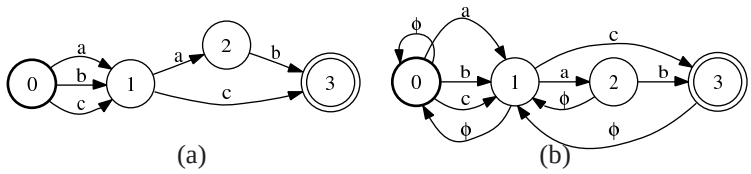


FIGURE 7 (a) Local grammar automaton A; (b) deterministic automaton B recognizing $\Sigma^* L(A)$ represented with failure transitions.

transitions.

The algorithm just described admits an on-the-fly implementation which makes it suitable for expanding only those states and transitions of the result need for the intersection with T.

An offline construction is preferable when multiple applications of the local grammar are expected. Unlike the algorithm presented in the previous section, the determinization of A' is then computed just once. The resulting automaton B' is compact thanks to the use of failure transitions.

The use of B' can be further optimized in a way similar to what can be done in the case of the algorithm of Knuth et al. (1977) using the following observation: if a label a is unavailable at q, it is also unavailable at the default state q' of q if the set of labels at q' is included in set of labels at q. Let the *context of* q be defined by:

$$C(q) = \{a \in \Sigma : \delta(q, a) \neq \emptyset\}.$$

To speed up the use of default transitions, the new transition function δ' can thus be defined as follows:

$$\delta'(q, \phi) \leftarrow \begin{cases} \delta(q, \phi) & \text{if } C(\delta(q, \phi)) \not\subseteq C(q) \text{ or } \delta(q, \phi) = q; \\ \delta'(\delta(q, \phi), \phi) & \text{otherwise.} \end{cases}$$

For example, the context of state 3 contains that of its default state 1 in the automaton of Figure 6(b). Thus, its default transition can be redefined to point to the default state of state 1, that is state 0.

Figures 7 and 8 provide a full example of application of a local grammar using the algorithm described. Figure 7(a) shows an example of a local grammar automaton A. The application of the algorithm produces the compact deterministic automaton B of Figure 7(b) represented with failure transitions.

Figure 8(a) shows a text automaton and Figure 8(b) the result of the application of the application of A to T obtained by intersecting B with T. The dotted transition is a transition not constructed during that intersection since it leads to the state pair $(2, 3)$ where 3 is a final state of B.

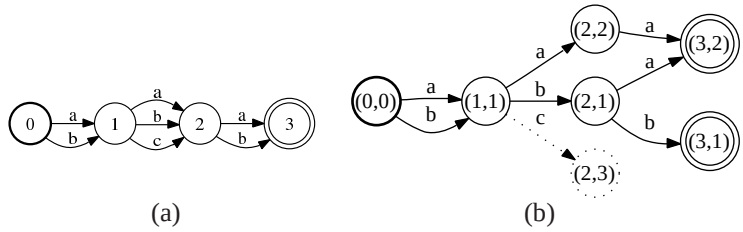


FIGURE 8 (a) Text automaton T; (b) Result of the application of the local grammar A to T.

9.3 Conclusion

Accurate local grammar automata are useful tools for disambiguation. They can significantly speed up the application of further text processing steps such as part-of-speech tagging or parsing. We gave a brief overview of several local grammar algorithms, including an efficient algorithm for their application to a text represented by an automaton.

Another natural way to define local grammars is to use context-dependent rewrite rules. Context-dependent rules can be efficiently compiled into finite-state transducers that can then be readily applied to an input text automaton (Kaplan and Kay, 1994, Mohri and Sproat, 1996). They can be further generalized to weighted context-dependent rules compiled into weighted transducers (Mohri and Sproat, 1996).

References

Aho, Alfred V. and Margaret J. Corasick. 1975. Efficient string matching: An aid to bibliographic search. *Communication of the Association for Computing Machinery* 18 (6):333–340.

Aho, Alfred V., Ravi Sethi, and Jeffrey D. Ullman. 1986. *Compilers, Principles, Techniques and Tools.* Addison Wesley: Reading, MA.

Allauzen, Cyril, Mehryar Mohri, and Brian Roark. 2005. The Design Principles and Algorithms of a Weighted Grammar Library. *International Journal of Foundations of Computer Science* (to appear).

Gross, Maurice. 1968. *Grammaire transformationnelle du francais.*, vol. 1, Syntaxe du verbe. Larousse.

Gross, Maurice. 1997. The Construction of Local Grammars. In *Finite-State Language Processing*, pages 329–354. The MIT Press, Cambridge, Massachusetts.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20(3).

Knuth, D.E., J.H. Morris, and V.R. Pratt. 1977. Fast Pattern Matching in Strings. *SIAM Journal on Computing* 6:323–350.

Koskenniemi, Kimmo. 1990. Finite-State Parsing and Disambiguation. In *Proceedings of the thirteenth International Conference on Computational Linguistics (COLING'90), Helsinki, Finland*.

Maurel, Denis. 1989. *Reconnaissance de séquences de mots par automates. Adverbes de date*. Ph.D. thesis, Université Paris 7.

Mohri, Mehryar. 1997a. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics* 23:2.

Mohri, Mehryar. 1997b. String-Matching with Automata. *Nordic Journal of Computing* 4:2.

Mohri, Mehryar and Richard Sproat. 1996. An Efficient Compiler for Weighted Rewrite Rules. In *34th Meeting of the Association for Computational Linguistics (ACL '96), Proceedings of the Conference, Santa Cruz, California*.

Roche, Emmanuel. 1992. Text disambiguation by finite state automata, an algorithm and experiments on corpora. In *Proceedings of COLING-92*.

10

Twol at Work

SJUR MOSHAGEN, PEKKA SAMMALLAHTI AND TROND TROSTERUD

10.1 Introduction

In this article, we will show two-level morphology at work. In sections 10.2 and 10.3, we will lay out the foundation for the work, by presenting the philosophy behind the Northern Saami two-level parser. In sections 10.4.1, 10.4.2 and 10.4.3 we then have a look at different applications, pedagogical programs, spell checking and terminology management.

10.2 Two-level morphology

Languages with ample morphophonological variation pose a problem for automatic analysis. A recapitulation of historical changes (or phonological rules) in two-level rules may be an elegant solution from the point of view of an overall grasp of the language in question but one soon runs into difficulties in dealing with products of analogical levellings and other exceptions, especially when text words consist of several morphemes each interacting with the other phonologically. One such language is Saami where word stems interacting with affixes can have over 20 phonological variants and derivational and inflectional morphemes interacting with word stems or each other more than five phonological variants.

After unsuccesfully trying different morphophonological rule approaches to the variation stemming from morpheme interaction an **indexed concatenation model** was devised. This model provides **phonological/graphemic morpheme variants** with indexes (or abstract features) according to their continuation categories, the sets of suffixes it precedes. In practice a phono-

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

logical/graphemic variant of a word stem occurring in front of a certain set of suffixes receives an index different from that of another variant occurring before a different set of suffixes.

Since the distribution of stem variants in relation to suffix sets vary from one stem type to another, a single phonological/graphemic variant of a word stem may belong to two or more morphophonological variants if a different stem has two or more phonological/graphemic variants before the same set of suffixes. Accordingly the suffixes are indexed according to their relations with morphophonological word stem variants.

The interaction with the word stems *giehta* 'hand; arm' and*njunni* 'nose' with the nominative plural suffix *Ât* and the second person singular accusative possessive suffix *Ât* may serve as an example. Both suffixes call for **weak grade** in the stem consonant center (the consonants between a stressed and a stressless vowel): (*njunni* →) *njuni-t* 'noses' and (*giehta* →) *gieđa-t* 'hands/arms'. However, the two suffixes call for different stem vowel alternants in i-stems but not in a-stems: *njuni-t* 'noses' ≠ *njuná-t* 'your nose' but *gieđa-t* 'hands/arms' = *gieđa-t* 'your hand/arm'. The stems *njuni-* and *njuná-* receive different indexes or abstract features (such as *njuni-N1* and *njuná-N2*) because they call for different sets of suffixes but so do the corresponding instances of the stem *gieđa-* (*gieđa-N1* and *gieđa-N2*) because the suffixes it precedes belong to two sets.

Since nouns have partly the same mophophonological variants as verbs, the morphophonological variants receive three kinds of indexes: N for the morphophonological variants of nouns, V for the morphophonological variants of verbs, and X for the morphophonological variants shared by nouns and verbs. To illustrate the use of indexes, a selection of noun and verb inflectional forms from three different parisyllabic stems are given. The examples represent types which show maximal morphophonological variation. Those with monophthongs in the first syllable (such as *ruhta* 'money', *lohti* 'wedge', *rohtu* 'grove'; *sihtat* 'want', *bihtit* 'to be strong enough for something', *vihkut* 'to suspect') or with other kinds of consonant centers (such as *gálgat* 'to undo', *máhttit* 'to know how', *riggut* 'to become rich') have a smaller number of morphophonological variants because some word stem accommodations such as diphthong simplification or extra strong grade do not manifest themselves in them. Cf. tables 1 and 2.

TABLE 1 Noun Stems: *giehta* ‘hand; arm’, *goahti* ‘hut; Saami tent’, *niehku* ‘dream’

NomSg	giehta-0	goahti-0	niehku-0
IllSg	gihti-i	goahtá-i	nihku-i
LocSg	gieđa-s	goađi-s	niegu-s
ComSg	gieđa-in	gođi-in	niegu-in
Ess	giehta-n	goahti-n	niehku-n
NomPl	gieđa-t	goađi-t	niegu-t
NomSg+Sg2Px	giehta-t	goahtá-t	nihko-t
GenSg+Sg2Px	gieđa-t	goađá-t	nigo-t

TABLE 2 Verb Stems: *viehkat* ‘to run’, *boahtit* ‘to come’, *biehkut* ‘to complain’

Inf	viehka-t	boahti-t	biehku-t
IndPrsSg1	viega-n	boađá-n	biegu-n
IndPrsSg3	viehká-0	boahtá-0	biehku-0
IndPrt Du3	viega-iga	bođi-iga	biegu-iga
IndPrtPl3	vihke-t	bohte-t	bihko-t
PotPrsSg1	viega-žan	bođe-žan	bigo-žan
CondPrsSg1	viega-šin	boađá-šin	bigo-šin
ImprtPrsSg2	viega-0	boađe-0	biego-0
ImprtPrsSg3	vihk-os	boht-os	bihk-os
ImprtPrsDu2	viehkki-0	boahtti-0	biehkku-0
ImprtPrsPl1	vihk-ot	boht-ot	bihko-t
VrbGen	viega-0	boađi-0	biegu-0
PrsPtcComSg	vihkki-in	bohtti-in	biehkku-in
PrfPrc	viehka-n	boahtá-n	bihko-n
PassiveStem	vihkk-o-	bohtt-o-	bihkk-o-

The inflectional morphemes are accompanied with a number of word stem accommodations, the ones relevant with regard to the examples in the paradigms are shown in table 3:

TABLE 3 Word Stem Accomodations

(a)	Weak Grade (WG):	hk → g, ht → đ
(b)	Extra Strong Grade (ESG):	hk → hkk, ht → htt
(c)	Diphtong Simplification (DS):	ie → i, oa → o, uo → o
(d)	Second Syllable Vowel Accomm. I (SSVAI):	i → á, u → o
(e)	Second Syllable Vowel Accomm. II (SSVAII):	i → e, u → o
(f)	Second Syll. Vowel Accomm. III (SSVAIII):	a → i
(g)	Second Syll. Vowel Accomm. IV (SSVAIV):	a → á, i → á
(h)	Second Syll. Vowel Accomm. V (SSVAV):	a → i, i → á
(i)	Second Syll. Allegro Shortening (SSAS):	i → e, u → o (+ á → a)
(j)	Second Syll. Vowel Loss Before Suffixal Vowel (SSVL):	-a → -0, -i → -0, -u → -0

We can see that the distribution of these accommodations, each of which corresponds to a rule in the two-level analysis program, is not the same in the paradigms representing different stem types. Grade Alternation, Consonant Center Strengthening, Second Syllable Allegro Shortening and Second Syllable Vowel Loss Before Suffixal Vowel occur in the same paradigmatic forms for all types of parisyllabic stems but Diphthong Simplification and the different Second Syllable Vowel Accommodations do not. On closer inspection, however, it becomes clear that DS and SSVAs are conditioned by the phonological properties of the stems and that their distribution differences depend on their applicability to different kinds of stems.

The stem variants can now be grouped according to the combinations of accommodations in the suffixes the continuation categories call for; if Diphthong Simplification is restricted to a stem type, it will be indicated with the stem vowel in braces. Out of the 15 groups of stem variants requiring different sets of suffixes in table 4, one is specific to nouns (nr. 4), 9 are specific to verbs (nrs 3, 5, 6, 9, 10, 12-15) and 6 are shared by nouns and verbs (nrs 1, 2, 6, 7, 8, 11). This means that in the presented examples, noun stems have 7 variants which require different continuation categories and verb stems have 13, as in table 4.

The continuation categories presented here are considerably simplified for the purposes of this paper; those in the actual program take various kinds of redundances as well as sequences of continuation categories into account and are far more complex. Furthermore, most of the continuation categories contain a number of suffixes in addition to those in the examples.

With the indexed concatenation model outlined here, it is possible to deal

TABLE 4 Stem variants

(1) **X1**: *No Accommodation*			
N+Sg+Nom: 0	giehta-0	goahti-0	niehku-0
N+Ess: n	giehta-n	goahti-n	niehku-n
V+Inf: t	viehka-t	boahti-t	biehku-t
(2) **X2**: *SSVAI*			
N+Sg+NomSg+Sg2Px: t	giehta-t	goahtá-t	nihko-t
V+PrfPrc: n	viehka-n	boahtá-n	bihko-n
(3) **V1**: *SSVAIV*			
V+Ind+Prs+Sg3: 0	viehká-0	boahtá-0	biehku-0
(4) **N1**: *DS(a,u) + SSVAV*			
N+Sg+Ill: i	gihti-i	goahtá-i	nihku-i
(5) **V2**: *DS + SSVAII*			
V+Ind+Prt+Pl3: t	vihke-t	bohte-t	bihko-t
(6) **V3**: *DS + SSVL*			
V+Imprt+Prs+Sg3: os	vihk-os	boht-os	bihk-os
V+Imprt+Prs+Pl1: ot	vihk-ot	boht-ot	bihk-ot
(7) **X3**: *WG*			
N+Sg+Loc: s	gieđa-s	goađì-s	niegu-s
N+Pl+Nom: t	gieđa-t	goađì-t	niegu-t
V+VrbGen: 0	viega-0	boađì-0	niegu-0
(8) **X4**: *WG + SSVAI*			
N+Sg+Gen+Sg2Px: t	gieđa-t	goađá-t	nigo-t
V+Cond+Prs+Sg1: šin	viega-šin	boađá-šin	bigo-šin
(9) **V4**: *WG + SSAS*			
V+Imprt+Prs+Sg2: 0	viega-0	boađe-0	biego-
(10) **V5**: *WG + SSVAIV*			
V+Ind+Prs+Sg1: n	viega-n	boađá-n	biegu-n
(11) **X5**: *WG + DS(i)*			
N+Sg+Com: in	gieđa-in	gođì-in	niegu-in
V+Ind+Prt+Du3: iga	viega-iga	bođì-iga	biegu-iga
(12) **V6**: *WG + SSVAII*			
V+Pot+Prs+Sg1: žan	viega-žan	bođe-žan	bigo-žan
(13) **V7**: *ESG + SSVAIII*			
V+Imprt+Prs+Du2: 0	viehkki-0	boahtti-0	biehkku-0
(14) **V8**: *ESG + DS(a,i) + SSVAIII*			
N+Sg+Com: in	vihkki-in	bohtti-in	biehkku-in
(15) **V9**: *ESG + DS + SSVL*			
V+Passive: o	vihkk-o-	bohtt-o-	bihkk-o-

with all kinds of complex morphophonologies in a straightforward manner. It is also relatively easy to add new morphophonological accommodations into the analyser.

10.3 Disambiguation

Disambiguation may be done in several ways. One is Finite-state intersection grammar, as suggested by Koskenniemi (1997). In our Saami project, we have chosen a different path, and use constraint grammar, as presented by Tapanainen (1996), here in Eckhard Bick's open-source version *vislcg* (cf. *sourceforge.net/projects/vislcg/*. This component is being written by Trond Trosterud and Marit Julien. Although still under development, it is already good enough to match the level of statistically-based POS taggers. At its present stage, it contains approximately 1300 disambiguation rules.

10.4 Twol in use

A grammatical analysator can be used for many purposes. We will here have a look at some areas where the Saami analysator has been put to use.

10.4.1 Pedagogical programs

The Saami analysator has been used to analyse sentences for interactive pedagogical syntax learning in the so-called visl project (Visual Syntax Learning) at Syddansk Universitet. The format in itself is not dependent upon having a grammatical analysator, but the analysator makes it possible to add sentences automatically. The process behind the analysis in Figure 3 consists of three parts:

1. The morphological analyser gives all possible analyses
2. A morphological disambiguator removes the incorrect ones, and adds syntactic functions
3. A phrase structure grammar gives the linear representation a hierarchical structure

At present, beta versions of the first two components are in place. The third component is still missing. For a sentence like *Áhčči lea oastán munnje divrras sabehiid* 'Father has bought me a an expensive pair of skis', the morphological analyser gives the representation in Figure 1.

After disambiguation and adding of syntactic functions, the same sentence can be seen in Figure 2.

The underlying representation of the pedagogical program takes the disambiguated analysis as input, and makes a tree structure, as seen in Figure 3. At present, this process is only partly automatised, the bracketing of constituents (denoted with '=') must be done manually, making such a component is the next task ahead.

```
"<Áhčči>"
         "áhčči" N Sg Nom
"<lea>"
         "leat" V Ind Prs Sg3
"<oastán>"
         "oastit" V PrfPrc
         "oastit" V* N Actor Sg Nom PxSg1
         "oastit" V* N Actor Sg Gen PxSg1
         "oastit" V* N Actor Sg Acc PxSg1
         "oasti" N Sg Nom PxSg1
         "oasti" N Sg Gen PxSg1
         "oasti" N Sg Acc PxSg1
"<munnje>"
         "mun" Pron Pers Sg1 Ill
"<divrras>"
         "divrras" A Attr
         "divrras" A Sg Nom
"<sabehiid>"
         "sabet" N Pl Gen
         "sabet" N Pl Acc
"<.>"
         "." CLB
```

FIGURE 1 Morphological analysis

```
"<Áhčči>"
        "áhčči" N Sg Nom @SUBJ
"<lea>"
        "leat" V Ind Prs Sg3 @+FAUXV
"<oastán>"
        "oastit" V PrfPrc @-FMAINV
"<munnje>"
        "mun" Pron Pers Sg1 Ill @ADVL
"<divrras>"
         "divrras" A Attr @AN>
"<sabehiid>"
        "sabet" N Pl Acc @OBJ
"<.>"
```

FIGURE 2 Disambiguated version

```
S:n('áhčči',sg,nom)    Áhčči
P:g
=D:v('leat',ind,pr,3sg) lea
=H:v('oastit',pcp2)     oastán
A:pron('mun',<pers>,1sg,ill)    munnje
Od:g
=D:adj('divrras',attr)  divrras
=H:n('sabet',pl,acc)    sabehiid
```

FIGURE 3 Underlying representation in the pedagogical program

A pilot set of 200 Saami sentences will shortly be included in the *http://visl.sdu.dk/visl/*. As soon as they become part of that pedagogical platform, the sentences will be reused in a large range of pedagogical programs, ranging from interactive syntax analysis via word-class shooting games to text analysis.

Later, when the analysator becomes better, it will also be possible to offer an open system, where the computer analyses user input and offers an interface for the user to analyse for himself.

10.4.2 TWOL as generator 1: paradigm generation for a terminological database

The bidirectional nature of the two-level model makes it ideal not only for analysis but also for generation of word forms. An example of this is seen in Figure 4, using the current North Saami transducer.

```
xfst[1]: apply up
apply up> máná
mánná+N+Sg+Acc
mánná+N+Sg+Gen

xfst[1]: apply down
apply down> mánná+N+Sg+Acc
máná
```

FIGURE 4 Analysis and generation of the same word form using the same two-level model in oposite directions

This feature of the two-level model will be put into use in a terminological database developed by the Saami parliament (at *http://www.risten.no/*) to generate complete paradigms at runtime of any entry in the database. The paradigm generation will first be implemented for North Saami, and later for Julev (Lule) Saami and other Saami languages.

Unless special attention is paid to homonym entry words with different inflections, the simple application of a two-level model in the oposite direction will overgenerate, leading to wrongly generated word forms. One way of dealing with the overgeneration would be to add a unique string to homonyms as part of their lexical entry, as shown in Figure 5. This would ensure round-trip consistency without over-generation (one will always be able to generate exactly what was analyzed), but would complicate generation of such homonyms if no analysis is available: without knowing that it is a homonym, how would one know that the word requires a special tag to be generated?

With appropriate rules for dealing with homonym indices like *_1* and *_2*, the analysis output (and hence the generator input) would include the necessary info to generate complete and accurate paradigms without overgenera-

```
beassi:beassi_1 GOAHTI ;
beas0s0i:beas'si_2 GOAHTI ;
ára:ára_1 GOAHTI ;
á0ra0:árran2_2 SEAMU ;
```

FIGURE 5 Homonym lexical entries with an additional differentiator to differentiate them in analysis and generation

tion.

10.4.3 Spell checker

Using two-level technologies for making spellers was early on a pretty obvious application of it (see Arppe (2002) for a summary of the different initiatives in Finland), and one way of implementing orthographic correction is briefly described in Beesley and Karttunen (2003). Commercial implementations have been available since 1986 from Lingsoft Oy for Finnish, then Swedish, later also several other languages. For languages with rich inflectional and/or derivational morphology and free compounding, like Saami languages as well as many others, using a two-level or similar approach is in practice the only possibility.

Two-level technology is not only good for spellers. For languages with free compounding, using two-level technologies can also improve hyphenation, cf. Karlsson (1985). A commercial implementation is available from Lingsoft Oy, and described at *http://www.lingsoft.fi/doc/d-finhyp9.html*.

Since October 2004 the Norwegian Saami Parliament has been running a project to create proofing tools for North and Julev (Lule) Saami. The project is based on the work by Pekka Sammallahti and the projects at the University of Tromsø, described earlier in this article.

Handling descriptive and normative models at the same time

The proofing tools project is using the same lexicons as other Saami language technology projects at the University of Tromsø. While the university projects by nature are descriptive and want to be able to analyse both standard orthography and common substandard variants, the proofing tools' goal is to help authors make written text conform to orthographic standards. The language of the proofing tools is thus a subset of the language of the other projects, and to create a two-level model for proofing tools, we need to extract this subset language.

A very simple, but often sufficient, way of doing this is to add a comment to unofficial variants, and remove these variants using text processing methods in a preprocessing stage before compiling the lexicon, typically using 'grep'. This is how it is implemented in the present version of the North Saami TWOL, cf Figure 6, where the comment *!SUB* marks substandard variants.

```
leans#mán0ni:leans#mán'ni VIVVA ;
lens#mán0ni:lens#mán'ni VIVVA ; !SUB
kapihtal GAHPIR ; !SUB
sektor GAHPIR ; !SUB
```

FIGURE 6 Example North Saami entries with comments for substandard forms

Another method would be to use the network operations available in current language technology tools such as the Xerox Finite State tools described in Beesley and Karttunen (2003). Using network subtraction it would be possible to remove from the full (descriptive) language model the language containing all and only a specified feature, e.g. +Sub. This would also imply that substandard variants that are morphophonological rather than lexical in their nature could easily be removed, which is not necessarily possible with the preprocessing approach discussed above.

Extending both the approaches briefly discussed above would also make it possible to cover dialectal variation. Comparative forms of North Saami adjectives have one standard variant used in western dialects, and another standard form used in eastern dialects. Each of these dialect groups has in addition another variant ending in -u, which is not part of the orthography, but used orally and thus sometimes showing up in print. All four variants are listed in Figure 7, in their *lexc* representation.

```
Lexicon EABBO/EAMOS_CONT
+Comp+W:eabbo     EABBU ;  ! Parallel form Standard. West
+Comp+E:ab'bo     EABBU ;  ! Parallel form Standard. East
+Comp+W+Sub:eabbu EABBU ;  ! Parallel form Not standard. West
+Comp+E+Sub:ab'bu EABBU ;  ! Parallel form Not standard. East
```

FIGURE 7 Dialectal variation of Comparative, both standard and substandard variants

We have introduced two more tags in the description above, *+W* and *+E*, to denote western and eastern dialects respectively. We have also used a lexc tag *+Sub* instead of a comment to identify substandard variants. Using any of the methods above we can extract any combination of these parameters, that is, including or excluding substandard variants, for eastern or western dialects. In a speller context, this can be used to e.g. create a more strict speller, by excluding forms not relevant for the user.

TWOL as generator 2: full form list generation for LT-weak spellers

The proofing tools project at the Saami parliament will create spellers and hyphenators for several applications on Windows, MacOS X and Linux, using several different speller engines. Some of these engines are dictated by the applications the project needs to make spellers for, for other applications it

has been a goal to reduce the dependency on commercial software as much as possible. The most likely such speller engine is Aspell (*http://aspell.net/*), a derivative of iSpell, which employs a simple one-level automaton model for its engine. Aspell is open source, freely available, and runs on all operating systems the project plans to support. In addition, it plugs in to a multitude of applications on all platforms. Aspell's only weakness is its limited descriptive power, that is - the missing *two* level, which makes it quite hard to write spellers for languages with complex morphophonology.

Recreating the linguistic model in the limited format of Aspell is not an attractive choice: we do not want to maintain two sets of source files, let alone make sure they are syncronised. Developing a good linguistic model is in itself a huge task, and it is imperative that we can, with reasonable effort, reuse what has already been done.

Thanks to the bidirectional nature of the two-level model, it is relatively easy today to circumvent the limitations of spellers like Aspell to a large degree. As long as a transducer is non-circular, we can use it as a generator to not only produce paradigms as described in Section 10.4.2, but to print out the whole language of the transducer. The circular points in our lexicon are marked up, which makes it trivial to extract a subset of it that is non-recursive with a simple *grep* command. This subset still contains all the non-circular word formation, such as derivation, plus all the inflection.

Though quite substantial, such a generated full-form list will of course not have satisfying lexical coverage unless the underlying lexicon source files are themself «complete». A *complete* lexicon does not exist, but to make the actual lexicon as close to the ideal one as possible within the scope of the proofing project, we will add to the lexicon all entries found in available written material[1], including all compounds, such that the total lexical coverage should be quite good. It still remains to be seen whether the result will be satisfying. The following criteria will be essential in evaluating whether we are successful:

- size of resulting binary dictionary file
- speed of the speller
- precision and recall of the speller
- relevance of given suggestions

Using a full-form list also defeats another benefit of automatons and transducers, namely the space-efficient construction of regular morphology in con-

[1]The proofing project is building a corpus of Saami texts together with the disambiguator project; the corpus will hopefully contain substantial parts of the total body of Saami texts written in modern orthography

tinuation lexicons. iSpell[2] and lately Aspell[3] support what is called "affix compression" using "affix lexicons", which is a limited implementation of continuation lexicons[4]. The Divvun project will look into whether it would be possible to generate such affix lexicons automatically from the generated full-form list, or at least with minimal effort maintain such affix lexicons. This will be especially important if the resulting binary lexicons turn out to be very large without affix compression.

10.5 Summing up

In this article we have tried to describe issues in the development of the North Saami two-level model, and some of the applications for which it has been, or is going to be put to work. With the resent projects for pedagogical software and proofing tools, it is possible to secure a place in the modern, digital world for small languages like North Saami by using language technology rooted in Koskenniemi (1983).

References

Arppe, Antti. 2002. No single path - finnish lessons in the commercialisation of language engineering research. *http://www.ling.helsinki.fi/filt/info/articles/arppe-en.shtml* .

Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite State Morphology*. Studies in Computational Linguistics. Stanford, California: CSLI Publications.

Karlsson, Fred. 1985. Automatic hyphenation of Finnish. In *Computational Morphosyntax, Report on Research 1981-84*, vol. 13 of *Publications of the Department of General Linguistics, University of Helsinki*.

Koskenniemi, Kimmo. 1983. *Two-level Morphology: A General Computational Model for Word-form Production and Generation*. Publications of the Department of General Linguistics, University of Helsinki. Helsinki: University of Helsinki.

Koskenniemi, Kimmo. 1997. Representations and finite-state components in natural language. In E. Roche and Y. Schabes, eds., *Finite-State Language Processing*, Language, speech and communication, pages 99–116. Massachusetts: MIT Press.

Tapanainen, Pasi. 1996. *The Constraint Grammar Parser CG-2*, vol. 27 of *Publications of the Department of General Linguistics, University of Helsinki*. Helsinki: University of Helsinki.

[2] http://fmg-www.cs.ucla.edu/fmg-members/geoff/ispell.html
[3] since version 0.60.
[4] see *http://aspell.net/man-html/Affix-Compression.html* for details

11

Two Notions of Parsing

JOAKIM NIVRE

The term *parsing*, derived from Latin *pars orationis* (parts of speech), was originally used to denote the grammatical explication of sentences, as practiced in elementary schools. The term was later borrowed into computer science and linguistics, where it has acquired a specialized sense in connection with the theory of formal languages and grammars. However, in practical applications of natural language processing, the term is also used to denote the syntactic analysis of sentences in text, without reference to any particular formal grammar, a sense which is in many ways quite close to the original grammar school sense.

In other words, there are at least two distinct notions of parsing that can be found in the current literature on natural language processing, notions that are not always clearly distinguished. I will call the two notions *grammar parsing* and *text parsing*, respectively. Although I am certainly not the first to notice this ambiguity, I feel that it may not have been given the attention that it deserves. While it is true that there are intimate connections between the two notions, they are nevertheless independent notions with quite different properties in some respects. In this paper I will try to pinpoint these differences through a comparative discussion of the two notions of parsing. This is motivated primarily by an interest in the problem of text parsing and a desire to understand how it is related to the more well-defined problem of grammar parsing. In a following companion paper I will go on to discuss different strategies for solving the text parsing problem, which may or may not involve grammar parsing as a crucial component.

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

S	→	NP VP PU	JJ	→	Economic
VP	→	VP PP	JJ	→	little
VP	→	VBD NP	JJ	→	financial
NP	→	NP PP	NN	→	news
NP	→	JJ NN	NN	→	effect
NP	→	JJ NNS	NNS	→	markets
PP	→	IN NP	VB	→	had
PU	→	.	IN	→	on

FIGURE 1 Context-free grammar for a fragment of English

11.1 Grammar Parsing

The notion of grammar parsing is intimately connected to the notion of a formal grammar G defining a formal language $L(G)$ over some (terminal) alphabet Σ. The *parsing problem* can then be defined as follows:

> Given a grammar G and an input string $x \in \Sigma^*$, derive some or all of the analyses assigned to x by G.

The analysis of formal grammars and their parsing problems goes back to the pioneering work of Noam Chomsky and others in the 1950's and continues to be a very active area of research. The most widely used formal grammar, both in computer science and in computational linguistics, is the *context-free grammar* (CFG) of Chomsky (1956). Figure 1 shows a context-free grammar defining a fragment of English including the sentence analyzed in Figure 2, which is taken from the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993).

Over the years, a variety of different formal grammars have been introduced, many of which are more expressive than the CFG model and motivated by the desire to provide a more adequate analysis of natural language syntax. This development started with the transformational grammars of Chomsky (1957, 1965) and has continued with syntactic theories like Lexical-Functional Grammar (Kaplan and Bresnan, 1982) and Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994). In recent years, there has been a special interest in so-called mildly context-sensitive grammars, exemplified by Tree-Adjoining Grammars (Joshi, 1985) and Combinatory-Categorial Grammar (Steedman, 2000), which appear to strike a good balance between linguistic adequacy and computational complexity. However, there has also been considerable interest in grammars that are less expressive but more efficient, notably frameworks based on finite-state techniques (cf. Koskenniemi, 1997).

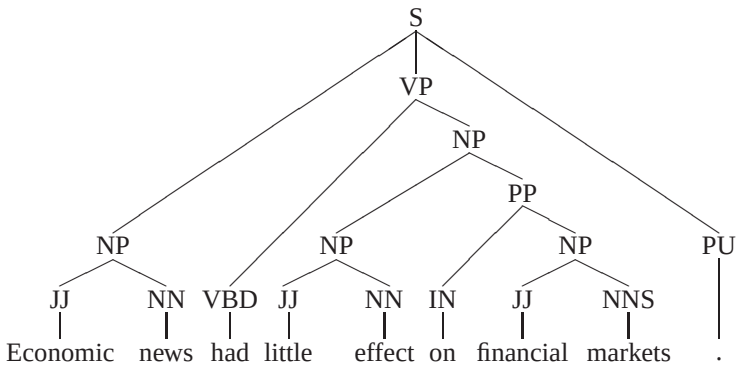


FIGURE 2 Constituent structure for English sentence

Solving the parsing problem for a specific type of grammar requires a parsing algorithm, i.e. an algorithm that computes analyses for a string x relative to a grammar G. Throughout the years a number of different parsing algorithms for different classes of grammars have been proposed and analyzed. For context-free grammars, some of the more well-known algorithms are the Cocke-Kasami-Younger (CKY) algorithm (Kasami, 1965, Younger, 1967), Earley's algorithm (Earley, 1970), and the left corner algorithm (Rosenkrantz and Lewis, 1970). These algorithms all make use of tabulation to store partial results, which potentially allows exponential reductions of the search space and thereby provides a way of coping with ambiguity. This type of method, which constitutes a form of *dynamic programming* (Cormen et al., 1990), can also be generalized to more expressive grammar formalisms.

Traditional parsing algorithms can be described as *constructive* in the sense that they analyze sentences by constructing syntactic representations in accordance with the rules of the grammar. An alternative to this is to use an *eliminative* parsing strategy, with treats the grammar as a set of constraints and views parsing as a constraint satisfaction problem. In this approach, which is found in different forms in frameworks such as Constraint Grammar (Karlsson, 1990, Karlsson et al., 1995), Parallel Constraint Grammar (Koskenniemi, 1990, 1997), and Constraint Dependency Grammar (Maruyama, 1990), sentences are analyzed by successively eliminating representations that violate constraints until only valid representations remain.

I will make no attempt to review the vast literature on grammar parsing here but will concentrate on some general observations concerning the properties of the parsing problem and the methods used to solve it. First of all, it is worth noting that the parsing problem for a class of grammars is a well-

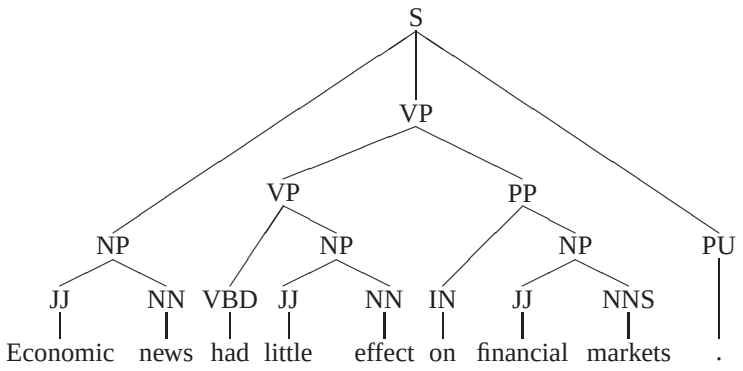


FIGURE 3 Alternative constituent structure for English sentence

defined *abstract problem* in the sense of algorithm theory (Cormen et al., 1990), i.e. a relation between a set I of inputs, which in this case are pairs consisting of a grammar G and a string x, and a set O of outputs, which are syntactic representations of strings in $L(G)$. A parsing algorithm provides a solution to this problem by computing the mapping from arbitrary inputs to outputs.

Secondly, the parsing problem for formal grammars is intimately tied to the corresponding *recognition problem*, i.e., the problem of deciding whether the string x is in $L(G)$. It is only strings in $L(G)$ that receive an analysis in the parsing process, and most parsing algorithms in fact solve the recognition problem simultaneously.

Thirdly, we note that the analyses to be assigned to a particular input string x are completely defined by the grammar G itself. For example, if G is a context-free grammar, we may be interested in the number of distinct parse trees that result from derivations of x from the start symbol S of G. In principle, this means that the correctness of a parsing algorithm can be established without considering any particular input strings, since the set of all input-output pairs are given implicitly by the grammar G itself.

The abstract nature of the grammar parsing problem is reflected in the evaluation criteria that are usually applied to parsing methods in this context. For example, a parsing algorithm is said to be *consistent* if, for any grammar G and input string x, it only derives analyses for x that are licensed by G; it is said to be *complete* if, for any G and x, it derives *all* analyses for x that are licensed by G. For example, the grammar in Figure 1 is ambiguous and assigns to our example sentence not only the analysis in Figure 2 but also

the analysis in Figure 3. Thus, a complete parsing algorithm must compute both these analyses, while a consistent algorithm must not compute any other analysis. However, both consistency and completeness can be proven without considering any particular grammar G or input string x, given the formal definition of the class of grammars and the relevant notions of derivation and representation.

The same goes for considerations of efficiency, where proofs of complexity, either for particular parsing algorithms or for classes of grammars, provide the most relevant tools for evaluation. For a context-free grammar G, parsing can be performed in $O(n^3)$ time, where n is the length of the input string x, using a dynamic programming algorithm. For mildly context-sensitive grammars, parsing complexity is still polynomial — typically $O(n^6)$ — while for more expressive formalisms running time becomes exponential in the worst case. By contrast, systems based on finite-state techniques normally support parsing in $O(n)$ time. Research on the complexity of linguistically motivated classes of grammars was pioneered by Barton et al. (1987) and has been followed by a wide range of subsequent studies.

Although complexity results often need to be supplemented by practical running time experiments, as shown for example by Carroll (1994), the role of empirical evaluation remains limited in grammar parsing, especially as far as correctness is concerned. This follows from the fact that grammar parsing is an abstract and mathematically well-defined problem, which can be studied using formal methods only.

11.2 Text Parsing

Text parsing[1] is concerned with the syntactic analysis of (more or less) unrestricted text. This notion of parsing therefore applies to concrete manifestations of a language L, where we cannot necessarily assume that L is a formal language. In particular, we are of course interested in the case where L is a natural language, or possibly a restricted subset of a natural language. I assume that a *text* in a language L is a sequence $T = (x_1, \ldots, x_n)$ of sentences (strings) x_i, and I define the *text parsing* problem as follows:

> Given a text $T = (x_1, \ldots, x_n)$ in language L, derive the correct analysis for every sentence $x_i \in T$.

The term *sentence* should be understood in the sense of *text sentence* rather than *system sentence* (Lyons, 1977), i.e., it refers to a segment of text with-

[1] The term *text* in *text parsing* is not meant to exclude spoken language, but rather to emphasize the relation to naturally occurring language use. Although I will have nothing to say about the parsing of spoken utterances in this paper, I want the notion of text parsing to encompass both written texts and spoken dialogues. An alternative term would be *discourse parsing*, but it seems that this would give rise to misleading associations of a different kind.

out any specific assumptions about syntactic completeness or other structural properties. What constitutes a sentence in this sense may differ from one language to the other and may not always be completely clear-cut. In the context of this paper I will simply disregard this problem, although it is well-known that the problem of sentence segmentation in text processing is far from trivial (Palmer, 2000).

To exemplify the notion of text parsing, let us return again to the example sentence from Figure 2. In its original context, which is a text taken from the Wall Street Journal and included in the Penn Treebank, this sentence has an interpretation that corresponds to the analysis in Figure 2 — rather than the alternative analysis in Figure 3. Therefore, the former analysis is the one and only correct analysis in the context of text parsing.

Let us now return to the observations made with respect to grammar parsing in the previous section and see in what respects text parsing is different. First of all, it is not clear that text parsing is a well-defined abstract problem in the same sense as grammar parsing, especially not when we consider texts in a natural language. It is true that text parsing has the structure of a mapping problem, but in the absence of a formal definition for the language L, there is no precise delimitation of the input set. Moreover, even if we can agree on the formal properties of output representations, there is no formal grammar defining the correct mapping from inputs to outputs. For example, the syntactic representation in Figure 2 is clearly of the kind that can be defined by a context-free grammar. But according to my conception of the text parsing problem, there is no specific instance of this formal grammar that defines the mapping from input strings to specific representations.

One way of looking at the problem is instead to say that it is an empirical *approximation problem*, where we try to approximate the correct mapping given increasingly large but finite samples of the mapping relation. Needless to say, this is a view that fits very well with a data-driven approach to text parsing, but the main point right now is simply that, unlike grammar parsing, the problem of text parsing lacks a precise characterization in formal terms.

Secondly, text parsing lacks the connection between parsing and recognition that we observed for grammar parsing. This is a direct consequence of the fact that the input language is not formally defined, which means that recognition is not a well-defined problem. Therefore, we can no longer *require* that an input string be part of the language to be analyzed. In most cases, we instead have to *assume* that any text sentence is a valid input string. And if we want to be able to reject some input strings as ill-formed, then we cannot refer to a formal language definition but must appeal to some other criterion.[2]

[2]For certain practical applications, such as grammar checking, it is obviously both relevant and necessary to reject certain strings by appeal to a prescriptive grammar, but it can be prob-

Thirdly, while there is no reference to a grammar in the definition of text parsing, there is reference to a sequence of sentences providing a textual context for each sentence to be analyzed. This is based on the assumption that text parsing deals with language use, and that the analysis assigned to a sentence is sensitive to the context in which it occurs. In particular, I assume that each text sentence has a single correct analysis, even if the string of words realizing the sentence may be found with other interpretations in other contexts. In other words, text parsing entails disambiguation.

However, the absence of a formal grammar also means that we need some external criterion for deciding what is the correct analysis for a given sentence in context. For natural languages, the obvious criterion to use is human performance, meaning that an analysis is correct if it coincides with the interpretation of competent users of the language in question. This leads to the notion of an *empirical gold standard*, i.e. a reference corpus of texts, where each relevant text segment has been assigned its correct analysis by a human expert. In the case of syntactic parsing, the relevant segments are sentences and the corpus will normally be a *treebank* (Abeillé, 2003, Nivre, 2005). Thus, my reason for saying that the analysis given in Figure 2 is correct is simply that this is the analysis found in the Penn Treebank.

The use of treebank data to establish a gold standard for text parsing is problematic in many ways, having to do both with the representativity of the corpus material and the reliability and validity of the treebank annotation. And even if we can establish a gold standard treebank, it will only provide us with a finite sample of input-output pairs, which means that any generalization to an infinite language will have to rely on statistical inference. This is in marked contrast to the case of grammar parsing, where the consistency and completeness of parsing algorithms, for any grammar and any input, can be established using formal proofs.

The empirical nature of the text parsing problem is reflected also in the evaluation criteria that are applied to parsing methods in this context. Since notions of consistency and completeness are meaningless in the absence of a formal grammar, the central evaluation criterion is instead the empirical notion of *accuracy*, which is standardly operationalized as agreement with gold standard data. However, it is important to remember that, even though it is often difficult to apply formal methods to the text parsing problem itself given its open-ended nature, the parsing methods we develop to deal with this problem can of course be subjected to the same rigorous analysis as algorithms for grammar parsing. Thus, if we are interested in the efficiency of different methods, we may use results about theoretical complexity of algorithms as well as empirical running time experiments. However, for the central notion

lematic in the general case.

of accuracy, there seems to be no alternative but to rely on empirical evaluation methods, at least given the current state of our knowledge.

11.3 Competence and Performance

The discussion of grammar parsing and text parsing leads naturally to a consideration of the well-known distinction between *competence* and *performance* in linguistic theory (Chomsky, 1965).[3] It may be tempting to assume that grammar parsing belongs to the realm of competence, while text parsing is concerned with performance. After all, the whole tradition of generative grammar in linguistics is built on the idea of using formal grammars to model linguistic competence, starting with Chomsky (1957, 1965). The idea that natural languages can be modeled as formal languages unites theorists as different as Chomsky and Montague (1970). Within this tradition, it might be natural to view the study of grammar parsing, when applied to natural language, as the study of idealized human sentence processing.

The traditional notion of linguistic competence has recently been called into question, and it has been suggested that many of the properties typically associated with linguistic performance, such as frequency effects and probabilistic category structure, also belong to our linguistic competence (Bod et al., 2003). While the nature of linguistic competence is a hotly debated and controversial issue, it seems unproblematic to assume that text parsing is concerned with performance, at least if we want to use text parsing methods to build systems that can handle naturally occurring texts. This means that a model of linguistic competence is of use to us only if it can be coupled with an appropriate model of performance. So, regardless of whether grammar parsing is a good model of linguistic competence or not, it is still an open question what role it has to play in text parsing (cf. Chanod, 2001).

11.4 Conclusion

The main conclusion that I want to draw from the discussion in this paper is that grammar parsing and text parsing are in many ways radically different problems and therefore require different methods. In particular, grammar parsing is an abstract problem, which can be studied using formal methods and internal evaluation criteria, while text parsing is an empirical problem, where formal methods need to be combined with experimental methods and external evaluation criteria. In a following companion paper I will discuss different methods that have been proposed for text parsing. Some of these methods crucially involve grammar parsing; others do not.

[3]Before Chomsky, similar distinctions had been proposed by Saussure (1916), between *langue* and *parole*, and by Hjelmslev (1943), between *system* and *process*, among others.

References

Abeillé, Anne, ed. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer Academic Publishers.

Barton, G. Edward, Robert C. Berwick, and Eric Sven Ristad. 1987. *Computational Complexity and Natural Language*. MIT Press.

Bod, Rens, Jennifer Hay, and Stefanie Jannedy, eds. 2003. *Probabilistic Linguistics*. MIT Press.

Carroll, John. 1994. Relating complexity to practical performance in parsing with wide-coverage unification grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 287–294.

Chanod, Jean-Pierre. 2001. Robust parsing and beyond. In J.-C. Junqua and G. van Noord, eds., *Robustness in Language and Speech Technology*, pages 187–204. Kluwer Academic Publishers.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* IT-2:113–124.

Chomsky, Noam. 1957. *Syntactic Structures*. Mouton.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*. MIT Press.

Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. 1990. *Introduction to Algorithms*. MIT Press.

Earley, J. 1970. An efficient context-free parsing algorithm. *Communications of the ACM* 13:94–102.

Hjelmslev, Louis. 1943. *Omkring sprogteoriens grundlæggelse*. Akademisk forlag.

Joshi, Aravind. 1985. How much context-sensitivity is necessary for characterizing structural descriptions – tree adjoining grammars. In D. Dowty, L. Karttunen, and A. Zwicky, eds., *Natural Language Processing: Psycholinguistic, Computational and Theoretical Perspectives*, pages 206–250. Cambridge University Press.

Kaplan, Ron and Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In J. Bresnan, ed., *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press.

Karlsson, Fred. 1990. Constraint grammar as a framework for parsing running text. In H. Karlgren, ed., *Papers presented to the 13th International Conference on Computational Linguistics (COLING)*, pages 168–173.

Karlsson, Fred, Atro Voutilainen, Juha Heikkilä, and Arto Anttila, eds. 1995. *Constraint Grammar: A language-independent system for parsing unrestricted text*. Mouton de Gruyter.

Kasami, T. 1965. An efficient recognition and syntax algorithm for context-free languages. Tech. Rep. AF-CRL-65-758, Air Force Cambridge Research Laboratory.

Koskenniemi, Kimmo. 1990. Finite-state parsing and disambiguation. In *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT)*, pages 6–9.

Koskenniemi, Kimmo. 1997. Representations and finite-state components in natural language. In E. Roche and Y. Schabes, eds., *Finite State Language Processing*, pages 99–116. MIT Press.

Lyons, John. 1977. *Semantics*. Cambridge University Press.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19:313–330.

Maruyama, Hiroshi. 1990. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Meeting of the Association for Computational Linguistics (ACL)*, pages 31–38. Pittsburgh, PA.

Montague, Richard. 1970. Universal grammar. *Theoria* 36:373–398.

Nivre, Joakim. 2005. Treebanks. In *Handbook of Corpus Linguistics*. Walter de Gruyter.

Palmer, David M. 2000. Tokenisation and sentence segmentation. In R. Dale, H. Moisl, and H. Somers, eds., *Handbook of Natural Language Processing*, pages 11–35. Marcel Dekker.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. CSLI Publications.

Rosenkrantz, D. J. and P. M. Lewis. 1970. Deterministic left corner parsing. In *Proceedings of the 11th Symposium on Switching and Automata Theory*, pages 139–152.

Saussure, Ferdinand de. 1916. *Cours de linguistique générale*. Payot.

Steedman, Mark. 2000. *The Syntactic Process*. MIT Press.

Younger, D. H. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control* 10:189–208.

12

Computational Morphologies for Small Uralic Languages

GÁBOR PRÓSZÉKY AND ATTILA NOVÁK

12.1 Introduction

This article presents a set of morphological tools for small Uralic languages. Various Hungarian research groups specialized in Finno-Ugric linguistics and a Hungarian language technology company (MorphoLogic) have initiated a project with the goal of producing annotated electronic corpora for small Uralic languages. The languages described include Mordvin, Udmurt (Votyak), Komi (Zyryan), Mansi (Vogul), Khanty (Ostyak), Tundra Nenets (Yurak) and Nganasan (Tavgi). Most of these languages are endangered, some of them are on the verge of extinction, so their documentation is an urgent scientific task. The most important subgoal of the project was to create morphological analyzers for the languages involved.[1]

In the project, we used the morphological analyzer engine called *Humor* ('High speed Unification MORphology') developed at MorphoLogic (Prószéky and Kis (1999)), which had been first successfully applied to another Uralic (Finno-Ugric) language, Hungarian, and later to various Slavic, Germanic and Romance languages. We supplemented the analyzer with two additional tools: a lemmatizer and a morphological generator. We present the tools through their application to the Komi language, specifically to the standard Komi-Zyryan dialect.

Creating analyzers for the two Samoyed languages involved in the project,

[1] The project was funded by the National Research and Development Programmes of Hungary ('Complex Uralic Lingustic Database', NKFP 5/135/2001).

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

Nenets and Nganasan, turned out to be a great challenge. Nganasan morphology and especially its phonology is very complex and the available linguistic data and their linguistic descriptions proved to be incomplete and partly contradictory, which made numerous revisions to our computational model necessary. Thus using the Humor formalism, which we successfully applied to other languages in and outside the project, was not feasible in the case of Nganasan, as shown in the second part of the present article. We used instead the regular relation calculus based toolset, *xfst* of Xerox to create the analyzer.

12.2 The Humor Tools

12.2.1 Features of the Morphological Analyzer

The Humor analyzer performs a classical 'item-and-arrangement' (IA) style analysis. The input word is analyzed as a sequence of morphs. It is segmented into parts which have (i) a surface form (that appears as part of the input string), (ii) a lexical form (the 'quotation form' of the morpheme) and (iii) a category label (which may contain some structured information or simply be an unstructured label). The lexical form and the category label together more or less well identify the morpheme of which the surface form is an allomorph.

The analyzer produces flat morph lists as possible analyses, since it contains a regular word grammar, which is represented as a finite-state automaton.

The following is a sample output of the Humor analyzer for the Komi word form *kylanly* ('to a listener/listening one').

```
analyzer>kylanly
 kyv[S_V]=kyl+an[D=A_PImpPs]+ly[I_DAT]
 kyv[S_V]=kyl+an[D=N_Tool]+ly[I_DAT]
```

Morphs are separated by `+` signs from each other. The representation of morphs is `lexical form[category label]=surface form`. A prefix in category labels identifies the morphological category of the morpheme (stem, derivational/inflectional suffix). In the case of derivational affixes, the syntactic category of the derived word is also given.

12.2.2 How the analyzer works

The program performs a search on the input word form for possible analyses. It looks up morphs in the lexicon the surface form of which matches the beginning of the input word (and later the beginning of the yet unanalyzed part of it). The lexicon may contain not only single morphs but also morph sequences. These are ready-made analyses for irregular forms of stems or suffix sequences, which can thus be identified by the analyzer in a single step, which makes its operation more efficient.

In addition to assuring that the requirement that the surface form of the

next morpheme must match the beginning of the yet unanalyzed part of the word (uppercase-lowercase conversions may be possible) is met, two kinds of checks are performed by the analyzer at every step, which make an early pruning of the search space possible.

On the one hand, it is checked whether the morph being considered as the next one is locally compatible with the previous one. On the other hand, it is examined whether the candidate morph is of a category which, together with the already analyzed part, is the beginning of a possible word construction in the given language. Possible word structures are described by an extended finite-state automaton.

12.2.3 The Lemmatizer

Our 'lemmatizer' tool, built around the analyzer core, does more than just identifying lemmas of word forms: it also identifies the exposed morphosyntactic features. In contrast to the more verbose analyses produced by the core analyzer, compound members and derivational suffixes do not appear as independent items in the output of the lemmatizer, so the internal structure of words is not revealed.

The analyses produced by the lemmatizer are well suited for such tasks as corpus tagging, indexing and parsing. The output of the lemmatizer and the analyzer is compared in the example below:

```
analyzer>kylanly
 kyv[S_V]=kyl+an[D=A_PImpPs]+ly[I_DAT]
 kyv[S_V]=kyl+an[D=N_Tool]+ly[I_DAT]
lemmatizer>kylanly
 kylan[N][DAT]
 kylan[A][DAT]
```

The lemmatizer identifies the word form *kylanly* as the dative of the noun or adjective (in fact: participle) *kylan* ('listener', 'listening one').

12.2.4 The Generator

The generator produces all word forms that could be realizations of a given morpheme sequence. The input for the generator is a lemma followed by a sequence of category labels which express the morphosyntactic features the word form should expose.

The generator is not a simple inverse of the corresponding analyzer, thus it can generate the inflected and derived forms of any multiply derived and/or compound stem without explicitly referring to compound boundaries and derivational suffixes in the input even if the whole complex stem is not in the lexicon of the analyzer. This is a useful feature in the case of languages where morphologically very complex stems are commonplace.

The following examples show how the generator produces an inflected form of the derived nominal stem *kylan*, which is not part of the stem lexicon, and the explicit application of the derivational suffix (and the same inflectional suffix) to the absolute verbal root of the word.

```
generator>kylan[N][DAT]
 kylanly
generator>kyv[V][_Tool][DAT]
 kylanly
```

It is possible to describe preferences for the cases when a certain set of morphosyntactic features may have more than one possible realization. This can be useful for such applications of the generator as text generation in machine translation applications, where the generation of a single word form is required.

12.3 The Morphological Database

Various versions of the Humor morphological analyzer have been in use for over a decade now. Although the analyzer itself proved to be an efficient tool, the format of the original database turned out to be problematic. For the analyzer to work efficiently, the data structures it uses contain redundant data. However, these redundant data structures are hard to read and modify for humans. So we built a morphological description development environment which facilitates the creation of the database.

12.3.1 Creating a Morphological Description

In the environment, the linguist has to create a high level human readable description which contains no redundant information and the system transforms it in a consistent way to the redundant representations which the analyzer uses. The work of the linguist consists of the following tasks:
a. Identification of the relevant morpheme categories in the language to be described (parts of speech, affix categories).
b. Description of stem and suffix alternations: an operation must be described which produces each allomorph from the lexical form of the morpheme for each phonological allomorphy class. The morphs or phonological or phonotactic properties which condition the given alternation must be identified.
c. Identification of features: all features (pertaining to the category or shape of morphemes, or to the idiosyncratic allomorphies triggered) playing a role in the morphology of the language must be identified.
d. Definition of selectional restrictions between adjacent morphs: selectional restrictions are described in terms of requirements that must be satisfied by the set of properties (features) of any morph adjacent to a morph. Each morph has two sets of properties: one can be seen by morphs adjacent to the left and

the other by morphs adjacent to the right. Likewise, any morph can constrain its possible neighbors by defining a formula expressing its requirements on each of its two sides.

e. Identification of implicational relations between properties of allomorphs and morphemes: these implicational relations must be formulated as rules, which either define how redundant properties and requirements of allomorphs can be inferred from their already known (lexically given or previously inferred) properties (including their shape), or define default properties.

f. Creation of stem and affix morpheme lexicons: in contrast to the lexicon used by the morphological analyzer, the lexicons created by the linguist contain the descriptions of morphemes instead of allomorphs. Morphemes are defined by listing their lexical form, category and all unpredictable features and requirements. A simple inheritance mechanism facilitates the consistent treatment of complex lexical entries (primarily compounds).

g. Creation of a word grammar: restrictions on the internal morphological structure of words (including selectional restrictions between nonadjacent morphemes) are described by a regular word grammar.

h. Creation of a suffix grammar (optional): a suffix grammar can be defined by setting up morphotactic classes for the suffixes and creating a directed graph labeled with the name these classes on its arcs. The development environment can produce segmented suffix sequences using this description and the suffix lexicon. Using such preprocessed segmented sequences enhances the performance of the analyzer.

As it can be seen from the description of the tasks above, we encourage the linguist to create a real analysis of the data (within the limits of the model that we provide).

12.3.2 Conversion of the Morphological Database

Using a description that consists of the information described above, the development environment can produce a lexical representation which already explicitly contains all the allomorphs of each morpheme along with all the properties and requirements of each of them. This representation still contains the formulae expressing properties and selectional restrictions in a human-readable form and can thus be easily checked by a linguist.

The readable redundant representation is then transformed to the format used by the analyzer using an encoding definition description, which defines how each of the features should be encoded for the analyzer.

12.4 The Komi Analyzer

In the subproject on Komi, which concentrates on the standard Komi-Zyryan dialect, we created a Komi morphological description using the development environment described in the previous section.

12.4.1 The Language

Komi (or Zyryan, Komi-Zyryan) is a Finno-Ugric language spoken in the northeastern part of Europe, West of the Ural Mountains. The number of speakers is about 300,000.

12.4.2 Creating a Komi Morphological Description

Since the annotated corpora we want to create are intended for linguists, we decided to use a quasi-phonological transcription of Komi based on Latin script instead of the Cyrillic orthography of the language. However, we plan to produce a Cyrillic version of the analyzer as well.

The first piece of description we created in the Komi subproject was a lexicon of suffix morphemes along with a suffix grammar, which describes possible nominal inflectional suffix sequences. One of the most complicated, though quite properly described, aspect of Komi morphology is the intricate interaction between nominal case and possessive suffixes.

A problem we were faced with was the lack of good and thorough modern synchronic grammars on many of the languages involved in the project. This was also the case for Komi, so we had to do a lot of research on the distribution of individual morphemes and allomorphies. In some cases we managed to get some information by producing the forms in question (along with their intended meaning) with the generator and having native speakers judge them.

An initial stem lexicon was created by hand using corpus data and a printed Komi-Russian dictionary (Beznosikova (2000)). Later we managed to acquire the dictionary in an electronic form. It contains about 31,000 stems plus 2800 names. Its conversion to the format used by the development environment is in progress.

There is a number of stem alternations in Komi. They are all triggered by attaching vowel initial suffixes. The alternations themselves are also very simple (there is an *l–v* alternation class and a number of epenthetic classes).

In many cases, it is predictable from the (quotation) form of a stem on phonotactic grounds whether it belongs to an alternation class. In other cases, this information must be entered into the stem lexicon. Since the underlying rules had not been described, finding them out was our task, and it is one of the scientific outcomes of the project.

12.5 Creating a morphological description for Nganasan

A formal description of Nganasan was written by fellow linguists taking part in the project (Wagner-Nagy (2002)). They also digitized a Russian-Nganasan dictionary (Kost'erkina et al. (2001)) and converted it to the phonemic transcription based on Latin script used by their team. The dictionary contains approximately 3,650 non-derived roots. The Nganasan team also provided

category labels for each item, which was missing from the original source. Wagner-Nagy (2002) also contains some short texts which we could use as a corpus along with a collection of text from other sources. Later we added another 500 roots encountered when testing the analyzer on this corpus.

During the preparation of the above described root dictionary, we also started to describe the suffixes of Nganasan in a formal manner. The first step of this was the creation of a list of the suffixes that contained the underlying phonological form of each suffix together with its category label, plus a feature that indicates which morphological root form the suffix can attach to. We used the following model to describe Nganasan morphology: we hypothesize that each root morpheme has three morphological stem variants (out of which two or all three might have the same form), and suffixes are sorted into three groups depending on which root allomorph they attach to. We also described the morphotactic restrictions governing the linear order of suffixes by defining a suffix grammar. The underlying phonological representation contains some archiphonemes: harmonic vowels and 'quasi-consonants' which never appear on the surface but condition gradation.

In Nganasan, nominal and verbal roots follow different alternation patterns. Additionally, vowel final and consonant final roots also exhibit different behavior. Some root-final changes are restricted to lexically marked root classes. Each of these roots must have a relevant lexical mark in the root inventory. Other root-final changes occur in each root satisfying the formal requirements of the rule.

12.6 The complexity of Nganasan morphophonology

It was relatively easy to describe root-final sound alternations in the Humor formalism. Those productive phonological processes that are sensitive to local contexts (such as degemination) could be formalized as separate rules. However, the phenomenon of gradation (i.e. the rule-governed alternation of obstruents in syllable onsets) proved to be so complex that we could not describe it satisfactorily. The root of the problem is that the Humor analyzer sees each word as a sequence of allomorphs and during analysis it checks whether the adjacent morphs are locally compatible with each other. Nganasan gradation, however, does not depend on the morphological make-up of the word: the only factor at play is syllable structure. Syllable boundaries and morph boundaries do not usually coincide. In the case of short suffixes (made-up of one segment), it is possible that even non-adjacent morphs belong to the same syllable. Moreover, the rules governing gradation in Nganasan are quite intricate. An obstruent in the onset position is in strong grade (i) in even-numbered open syllables (if not preceded by a long vowel) and (ii) if it is preceded by a non-nasal coda consonant. Otherwise, it is in rhythmical weak

grade (i) if preceded by a long vowel or (ii) if it is in odd-numbered syllable. Otherwise, it is in syllabic weak grade in even-numbered closed syllables. Gradation combines with other alternations in the language: vowel harmony, degemination, root alternations and various morphophonological suffix alternations (as a result of which a monosyllabic suffix can have as many as 20 different allomorphs).

To illustrate the complexity of the above outlined system let us look at the allomorphs of a single verbal suffix (of narrative mood used in the subjective and the non-plural objective conjugations). The underlying representation of the morpheme is `hA2nhV`, and its 12 allomorphs are: *banghu, bjanghy, bambu, bjamby, bahu, bjahy, hwanghu, hjanghy, hwambu, hjamby, hwahu, hjahy*. These allomorphs are produced from the underlying representation by the general phonological processes of the langage, undergoing vowel harmony, *a*-diphthongization and gradation.

While gradation is extremely difficult to formalize as a set of allomorph adjacency restrictions, it is such a productive process in Nganasan that it must be included in a proper morphological analyzer. It seemed, however, that though the formalism of the Humor analyzer proved to be adequate for the description of most phenomena in the language, the rule-formalism of the development environment could not cover all of the essential processes.

12.7 The application of a new formalism

In June 2003, a book was published (Beesley and Karttunen (2003)) with a CD containing a version of the two level morphological toolset of Xerox. This program set is based on finite state transducer technology and the versions published with the book can be freely used for non-commercial purposes. We decided to rewrite our description of Nganasan in the format used by the Xerox programs: lexc (Lexicon Compiler) and xfst (Xerox Finite-State Tool).

Using the xfst formalism, we could create a full description of Nganasan. The calculus implemented by the program makes it possible to ignore irrelevant symbols (such as morpheme boundaries in the case of gradation) in the environment description of re-write rules, therefore environments encompassing non-adjacent morphemes can be easily defined. As during composition the program automatically eliminates intermediate levels of representation created by individual rules producing a single finite-state transducer, generation and analysis can be performed efficiently.

Nganasan gradation was described in xfst as a cascade of rules performing syllabification, the identification of syllable grades, changing the quality of the obstruents in syllable onsets and removing auxiliary symbols. The rule system covers the irregularities of Nganasan syllabification. The whole of the rule system naturally contains several other rules. It describes all pro-

ductive, automatic phonological rules (e.g. the assimilation of nasals to the immediately following obstruent, degemination, vowel harmony, nunnation, palatalization etc.) and morphologically or lexically constrained root and suffix alternations.

We converted our morpheme inventories into the format used by lexc. Some of the feature-based constraints of the Humor description (e.g. the morphological stem selection) were retained in the new formalism: we used the 'flag diacritics' construct of the Xerox tools to implement them.

12.8 Conclusion

In addition to the ones described above, analyzers for Udmurt, Mari and Tundra Nenets have been finished.[2] The former two were prepared using the Humor based formalism, the latter was implemented using xfst and lexc. Additional analyzers for Mansi, Khanty and Mordvin are under construction using the Humor formalism.

A very important result of the project besides creating the programs and annotated corpora using them is that many gaps, uncertainties and inconsistencies were detected and in many cases corrected in the written grammars of these languages. Many details of the description which often remain vague in written grammars (such as the ordering and exact formulation of rewrite rules) must unavoidably made explicit in a computationally implemented grammar. Moreover, the adequacy of the implemented grammar can be very thoroughly tested against a great amount of real lingustic data. Systematic comparison of word forms generated against model paradigms has pinpointed errors not only in the computational implementation (which were then eliminated) but also in the model paradigms or the grammars the computational implementation was based on. We consider it very important to provide feedback to the linguists having prepared the original grammars and to publish the linguistic results of the project. We also hope that the many questions which remained open will induce further field research concerning these endangered languages and that they will be answered before it is too late.

References

Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite State Morphology*. Ventura Hall: CSLI Publications.

Beznosikova, Ljucija, ed. 2000. *Komi-Roča Kyvčukör (Komi-Russian Dictionary)*. Syktyvkar: Komi kniéznoe izdat.

[2]The individual analyzers were created by Attila Novák in co-operation with László Fejes (Komi, Udmurt, Mari), Beáta Wagner-Nagy and Zsuzsa Várnai (Nganasan) and Nóra Wenszky (Tundra Nenets). The Tundra Nenets analyzer is based on Tapani Salminen's work (Salminen (1997) and Salminen (1998), which he kindly made available to us in a machine readable form) and was created in close on-line co-operation with him.

Kost'erkina, N. T., A. Č. Momd'e, and T. Ju. Ždanova. 2001. *Slovar' nganasansko-russkij i russko-nganasanskij*. Sankt-Pet'erburg: Prosvesčen'ije.

Prószéky, Gábor and Balázs Kis. 1999. A unification-based approach to morpho-syntactic parsing of agglutinative and other (highly) inflectional languages. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 261–268. College Park, Maryland, USA.

Salminen, Tapani. 1997. *Tundra Nenets inflection*. Helsinki: Mémoires de la Société Finno-Ougrienne 227.

Salminen, Tapani. 1998. *A morphological dictionary of Tundra Nenets*. Helsinki: Lexica Societatis Fenno-Ugricae 26.

Wagner-Nagy, Beáta, ed. 2002. *Chrestomathia Nganasanica*. Szeged – Budapest: SZTE Finnugor Tanszék – MTA Nyelvtudományi Intézet.

13

Morphology and the Harappan Gods

RICHARD SPROAT AND STEVE FARMER

13.1 Introduction

Kimmo Koskenniemi has done work in a variety of areas having to do with the computational modeling of language, including computational syntax, information retrieval and, most famously, computational morphology. It is this latter area, and one other perhaps less well-known one, that are the topic of this chapter.

Koskenniemi's thesis work on the computational modeling of Finnish morphology (Koskenniemi, 1983) is certainly the best-known work in the field of computational morphology, and it has inspired a wealth of derivative work, including practical working morphological analyzers for a wide variety of languages.

One of his lesser known contributions is in the area of decipherment, namely his collaboration with the Finnish Indologist Asko Parpola on the computational analysis of the inscriptions of the Indus Valley.

In this chapter we will review these two contributions and their importance for their respective fields. Note that the first author of this paper may possibly be the only other person in the world who, like Koskenniemi, has done work on these two topics. The second author is the first author's collaborator on the Indus Valley work.

13.2 Koskenniemi's Contributions in Morphology

Koskenniemi's development of Two-Level Morphology can be thought of as a fortuitous accident of history. It had been known since C. Douglas Johnson's PhD thesis 1972 that "context-sensitive" rewrite rules of the kind that

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

had become familiar in generative phonology described regular relations and could thus be implemented using finite-state transducers (FSTs). By the late 1970's Ron Kaplan and Martin Kay at Xerox PARC were developing algorithms for the automatic compilation of FSTs from rewrite rules in a format that would be familiar to linguists, namely:

$$\phi \rightarrow \psi / \lambda \_\_ \rho \tag{13.1}$$

Here, ϕ, ψ, λ and ρ could be arbitrary regular expressions. Furthermore, since regular relations are closed under composition, this meant that one could write a series of ordered rules of the kind found in SPE (Chomsky and Halle, 1968), compile each of the rules into a transducer and then compose the entire series of rules together to form a single transducer representing the entire rule system. Kaplan and Kay finally published their algorithm many years later (Kaplan and Kay, 1994), and there has been subsequent work on a simpler and more efficient algorithm in Mohri and Sproat (1996).

But in the late 1970's and early 1980's there was just one problem: computers were simply not fast enough, nor did they have enough memory to compile rule systems of any serious complexity. Indeed complex rule systems of several tens of rules over a reasonable-sized alphabet (say 100 symbols) can easily produce FST's with several hundred thousand states with a similar number of arcs, with a total memory footprint of several megabytes. While any PC today could easily handle this, this was simply not viable around 1980.[1]

13.2.1 The two-level morphological system

Koskenniemi therefore proposed an alternative, one that still used transducers but constructed them and used them in a different way. First of all, he eschewed rule compilation entirely, instead constructing his transducers by hand. This is not quite as bad as it seems, since he proposed various ergonomically reasonable devices, such as the use of a "wildcard" ('=' in his notation) that would match any character not already mentioned: thus for any state, one could specify transitions to other states on designated symbol pairs, and have a default transition on '=:=' if none of the other specifications matched. This allowed the FSTs in Koskenniemi's description to be quite compact.

Second, rather than deal with rule composition, he came up with a novel alternative: the FSTs would run in parallel, each of them reading characters from the surface tape (the form of the word that appears in text) and the lexical tape (the form of the word that is entered in the lexicon, along with its morphosyntactic features). This presents a theoretical problem though, because a system of this kind is implementing *intersection* of FSTs and hence regular

[1] Recall Bill Gates' 1981 statement that "640k ought to be enough for anybody."

relations, whereas it is known that regular relations are not generally closed under intersection (Kaplan and Kay, 1994); however so long as the number of insertions or deletions is bounded, it can be shown that regular relations are closed even under intersection (Roark and Sproat, 2006), and in effect this is what Koskenniemi's system is doing when it constrains the transducers from getting too out of sync.

Koskenniemi's implementation of the lexical entries themselves, as well as affixes was less of an innovation. For the lexicons, he used the idea of letter *tries*, from Knuth (1973). To handle morphological decomposition he used the notion of *continuation lexicon* where a lexical entry would be annotated with information on what other lexical entries (usually affixes) could follow it. But this is just an implementation of a finite-state grammar and in fact Koskenniemi's trie-plus-continuation-lexicon approach is formally equivalent to representing the lexicons as finite-state acceptors (FSAs).

In Koskenniemi's original formulation, the input (surface) word would be matched against the lexicon by starting at a root lexicon and then matching the characters of the input against the characters in the lexicon trie, modulated by the parallel phonological transducers, which Koskenniemi picturesquely describes as viewing the lexicon through a slightly distorting lens. A present day two-level system would, of course, implement the following set of finite-state operations, where I is the input word, R_i are the rule transducers, and L is a lexical FSA:

$$I \circ \bigcap_i (R_i) \circ L \tag{13.2}$$

13.2.2 Two-Level Rules

The other innovation of Koskenniemi's approach was his formalization of two-level rewrite rules; again, he did not provide a compiler for these rules, but the rules served to specify the semantics underlying the transducers that he built by hand. All rules in his system followed a template in that they were all of the following form:

CorrespondencePair **operator** LeftContext __ RightContext

That is, the rules specified conditions for the occurrence of a correspondence pair — a pairing of a lexical and a surface symbol (one of which might be empty), modeling deletion or insertion — in a given left or right context. The contexts could be regular expressions, but the correspondence pair was a single pair of symbols, and thus was not as general as the $\phi \rightarrow \psi$ formulation from Kaplan and Kay (1994).

Koskenniemi's rules came in four flavors, determined by the particular **operator** used. These were:

Exclusion rule	a:b /⇐ LC _ RC
Context restriction rule	a:b ⇒ LC _ RC
Surface coercion rule	a:b ⇐ LC _ RC
Composite rule	a:b ⇔ LC _ RC

The interpretation of these was as follows:

- **Exclusion rule**: *a* cannot be realized as *b* in the stated context.
- **Context restriction rule**: *a* can only be realized as *b* in the stated context (i.e. nowhere else)
- **Surface coercion rule**: *a* must be realized as *b* in the stated context.
- **Composite rule**: *a* is realized as *b* obligatorily and only in the stated context.

In many ways the semantics of Koskenniemi's rules was better defined than the ones that had previously been used in generative phonology. For one thing, each rule type specified a direct relation between the underlying and surface forms, something that was not possible within generative phonology due to the arbitrary number of ordered rewrite rules: in general, in generative phonology there was no way to know how a given lexical form would surface, short of applying all rules in the specified order and seeing what the outcome was. Koskenniemi's rules, in contrast, specified the relation directly.

Ignoring for the moment that traditional generative phonological rules were not two-level, one can ask which of Koskenniemi's rules correspond to the rule types (basically just obligatory or optional rewrite rules) of generative phonology. In fact only the **surface coercion rule** has a direct counterpart: it corresponds pretty directly to an obligatory rewrite rule. All the other two-level rule types depend upon global knowledge of the system. Thus the **context restriction rule** is equivalent to a situation in a traditional generative account where there is but one optional rule that changes *a* into *b*; but note that this is a property of the system, not of a specific rule. The **composite rule**, which is just a combination of **context restriction** and **surface coercion** is similar, but in this case the unique rule changing *a* into *b* is obligatory. Note that since one could write, say, a **context restriction** rule that relates *a* to *b* in one environment, and then also write another **context restriction** rule that allows *a* to become *b* in another environment, it is perfectly possible in Koskenniemi's system to write an inconsistent grammar. A lot of the work in designing later two-level systems involved writing debuggers that would catch these kinds of conflicts. Finally, the **exclusion rule** is again global in nature: it is equivalent to the situation in a traditional generative grammar where there is no rule that relates *a* to *b* in the specified environment.

But really, Koskenniemi's rules can best be thought of as involving constraints on correspondence pairs. Constraints were virtually non-existent as a device in early generative phonology, but have since become quite popular

in various theories of phonology including Declarative Phonology (Coleman, 1992), One-Level Phonology (Bird and Ellison, 1994) and Optimality Theory (Prince and Smolensky, 1993).

13.2.3 Koskenniemi's impact on computational morphology

Koskenniemi's two-level morphology was remarkable in another way: in the early 1980's most computational linguistic systems were toys. This included parsers, which were usually fairly restricted in the kinds of sentences they could handle; dialog systems, which only worked in very limited domains; and models of language acquisition, which were only designed to learn simple grammatical constraints. In contrast, Koskenniemi's implementation of Finnish morphology was quite real in that it handled a large portion of inflected words that one found in real Finnish text. To some extent this reflects the fact that it is easier to get a quite complete coverage of morphology in any language than it is to have a similar coverage of syntax, let alone dialog. But it also reflects Koskenniemi's own decision to develop a full-fledged system, rather than present a mere "proof of concept" of his ideas.

While two-level morphology was originally motivated by the difficulties, at the time, with Kaplan and Kay's approach to cascaded rewrite rules, the model quickly took on a life of its own. Koskenniemi took it to be a substantive theoretical claim that only two levels of analysis were necessary, a claim that was fairly radical in its day (at least in contrast to generative phonology), but which has since been superseded by claims that only one-level is needed (e.g. Bird and Ellison, 1994).

Nevertheless, practical considerations of developing morphological analyzers have led people to not rely wholly on the two-level assumption. Since transducers can be combined both by composition (under which they are always closed) and by intersection (under which they are closed under certain conditions) combinations of these two operations may be used in any given system; see, e.g., Karttunen et al. (1992). Indeed, one of the beauties of finite-state techniques is that the calculus of the combination of regular languages and relations is expressive enough that one can develop modules of systems without regard to following any particular overall design: thus, for handling certain phenomena it may be more convenient to think in terms of a two-level system. For others, it may be easier to write cascaded rules. No matter: the two components can be combined as if one had built them both in one way or the other.

While Koskenniemi certainly did not invent finite-state approaches to morphology and phonology, he was the first to develop a system that worked fully using finite-state techniques, and he is thus to be given much credit for bringing the field of finite-state morphology to maturity, and building the way for the renaissance of finite-state approaches to language and speech that has de-

veloped over the past couple of decades.

13.3 Koskenniemi's Contributions to Indus Valley Studies

Koskenniemi's other contribution of interest here is his collaboration with the Indologist Asko Parpola in attempts to decipher the so-called Indus script (Koskenniemi and Parpola, 1980, 1982, Koskenniemi, 1981, Parpola, 1994). One of the products of that collaboration was the development of a concordance of Indus inscriptions (Koskenniemi and Parpola, 1979, 1982) that expanded on earlier work by Parpola and Koskenniemi's brother Seppo. Later on we will say a bit about that concordance, whose structure relies on the traditional assumption that the Indus symbols were part of a writing system, which we have recently challenged on a variety of statistical and non-statistical grounds (Farmer et al., 2004). Of deeper interest in the context of this paper is Koskenniemi's work on the automatic derivation of groupings among Indus symbols, which (on the linguistic assumption) he links to putative syntactic structures and to the detection in the inscriptions of possible homographs. Koskenniemi uses two main methods to distinguish sign groupings in the inscriptions. The first he attributes to S. Koskenniemi et al. 1970. The method, as Kimmo Koskenniemi describes it, involves comparing the actual counts of paired symbols with the expected counts based on the general frequencies of each sign. Symbol pairs with higher ratios are assumed to reflect underlying syntactic regularities in the system. This measure is related to pointwise mutual information (Shannon, 1948), which has been used extensively in computational linguistics for computing associations between words; for example, the measure was used in (Sproat and Shih, 1990) for the unsupervised discovery of word boundaries in Chinese texts and for parsing more generally in (Magerman and Marcus, 1990). Unfortunately, mutual information does not provide a solid foundation for syntactic analysis since high mutual information between terms is more often indicative of semantic association than syntactic constituency. While strong syntactic associations are sometimes found with closely linked terms, strong semantic associations also show up between terms that have no necessary syntactic relationship, e.g. between the English words *doctor* and *nurse*. Moreover, strong pairwise associations also show up often in non-linguistic strings, as witnessed in mathematical equations or chains of non-linguistic symbols associated with pantheons of gods (see (Farmer et al., 2004)), that have nothing to do with linguistic syntax.

Koskenniemi's other method ultimately derives from the work of Zellig Harris (1951). Starting from the left or right end of a sequence of glyphs, one counts, for each initial substring, the number of other texts that share the same beginning or end. One expects the number of possible next signs

to rise at a major syntactic boundary, since there are fewer restrictions across constituents than within constituents. Harris originally used essentially the same measure to determine the location of morph boundaries in unsegmented sequences of text. Koskenniemi argues that the two methods — the mutual-information-like method and the Harrisian method — produce similar syntactic analyses.

Koskenniemi also associates with Harris his method for detecting potential homographs. As Koskenniemi correctly notes, early writing systems were replete with homography, so it is reasonable to expect that Indus signs (based again on the assumption that they are linguistic) would also contain many homographs. The discovery of homographs is one of the trickiest aspects of decipherment. Based on what we know of the extensive homography of early scripts, we certainly cannot assume that a particular sign always has the same value; but at the same time we cannot simply assign homographs at will since such a strategy permits an unlimited number of potential decipherments of a given inscription with no obvious way to choose between them. Many of the well over 100 claimed decipherments that have been proposed in the past of the so-called Indus script have been plagued by this problem. The result is that a robust, replicable method for detecting potential homographs would be a useful tool in helping to select between potential linguistic readings of an undeciphered script. The method that Koskenniemi proposes to deal with this problem can be summarized as follows: consider symbols y and z, which occur in distinct linguistic environments, e.g. in two differing sets of preceding and following glyph environments. Now suppose one finds a glyph x that occurs in both of these environments: since x behaves in some cases like y and in other cases like z, x is a reasonable candidate for being a homograph. In other words, it is possible in this context that x is being used to represent two distinct linguistic entities. To provide an example from English, consider the words *carp* and *violin*. If one examines a corpus of English, one will likely find that the linguistic environments in which the word *carp* shows up have little in common with those that include the word *violin*. Now consider the word *bass*. If one looks again at the corpus, one will find that *bass* occurs both in environments similar to those in which *carp* appears and in environments similar to those in which we find *violin*. From this one can guess that *bass* is a potential homograph with two very different senses — in this case involving fish and musical instruments. A more sophisticated approach to automatic ambiguity detection along the lines of what Koskenniemi proposed, following Harris, was explored in (Sproat and van Santen, 1998).

The two problems that Koskenniemi addressed — the automatic detection of syntactic structures and of potential homographs — are topics that remain at the forefront of computational linguistics, as researchers search for more powerful automatic method of analyzing linguistic data. Unfortunately,

Koskenniemi's proposed methods have not had a major impact on Indus research, not due necessarily to any formal flaws in those methods, but instead, as suggested earlier, since those methods overlay the deeper, unexamined, assumption that Indus inscriptions encoded natural language. Non-linguistic sign systems often display levels of formal structure no less extreme than those seen in linguistic systems: witness the complex syntactic structures in mathematical expressions, or the recurrent sign groups that regularly show up in non-linguistic sign systems in the ancient Near East (Farmer et al., 2004). It has also long been known that non-linguistic signs display semantic "multivocality" that can be loosely pictured as the non-linguistic equivalent of homography in scripts. The upshot is that while Koskenniemi's methods may in fact identify genuine systematic relationships between symbols in Indus inscriptions, relationships of this type are not unique to writing but show up as well in a much wider class of sign systems. Since ethnographical studies suggest that the intended sense of nonlinguistic symbols are typically less "fixed" than those in written systems (cf., e.g., Barth (1987)), this finding also raises questions about the utility of the types of concordances of Indus inscriptions to which Koskenniemi has contributed, which overly standardize signs in ways that may mask important visual clues to the original sense of those signs, which may have differed widely in different Indus sites and periods as well as on diverse artifact types.

It is noteworthy that no unsupervised means has ever been proposed to distinguish linguistic from non-linguistic strings. It would be interesting to see whether the methods that Koskenniemi introduced in his studies of Indus signs might be applied to this interesting and still undeveloped area of research, grounded perhaps on systematic comparison of the specific types of regularities found in a significant cross-section of different classes of linguistic and non-linguistic sign systems. Those methods may also have possible applications in future studies of Indus symbols that are not tied to the traditional assumption, which is now being seriously challenged, that Indus inscriptions systematically encoded speech.

13.4 Summary

Koskenniemi has made many contributions to many areas in computational linguistics. This paper has reviewed what is certainly his best known contribution — two-level computational morphology, and what may well be his least-known contributions, namely his work on the Indus Valley corpus. Two-level morphology has, of course, been highly influential and is still used despite the fact that one of the main motivations for this approach (the processing power of early 1980's computers) is no longer relevant. Koskenniemi's work on the Indus Valley corpus is also interesting since, although we believe

there is compelling evidence that the Indus Valley "script" did not encode a language, he was investigating issues — the automatic discovery of structure, and the automatic discovery of senses — which are very much relevant today.

References

Barth, Frederik. 1987. *Cosmologies in the Making: A Generative Approach to Cultural Variation in Inner New Guinea*. Cambridge: Cambridge University Press.

Bird, Steven and T. Mark Ellison. 1994. One-level phonology: Autosegmental representations and rules as finite automata. *Computational Linguistics* 20(1):55–90.

Chomsky, Noam and Morris Halle. 1968. *The Sound Pattern of English*. New York: Harper and Row.

Coleman, John. 1992. *Phonological Representations — their Names, Forms and Powers*. Ph.D. thesis, University of York.

Farmer, Steve, Richard Sproat, and Michael Witzel. 2004. The collapse of the Indus-script thesis: The myth of literate Harappan civilization. *Electronic Journal of Vedic Studies* 11(2).

Harris, Zellig. 1951. *Methods in Structural Linguistics*. Chicago: University of Chicago Press.

Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description*. Mouton, The Hague: Mouton.

Kaplan, Ronald and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20:331–378.

Karttunen, Lauri, Ronald Kaplan, and Annie Zaenen. 1992. Two-level morphology with composition. In *COLING-92*, pages 141–148. COLING.

Knuth, Donald. 1973. *The Art of Computer Progamming*, vol. 3. Reading, MA: Addison-Wesley.

Koskenniemi, Kimmo. 1981. Syntactic methods in the study of the Indus script. *Studia Orientalia* 50:125–136.

Koskenniemi, Kimmo. 1983. *Two-Level Morphology: a General Computational Model for Word-Form Recognition and Production*. Ph.D. thesis, University of Helsinki, Helsinki.

Koskenniemi, Kimmo and Asko Parpola. 1979. Corpus of texts in the Indus script. Tech. rep., University of Helsinki. Department of Asian and African Studies, Research Reports 1.

Koskenniemi, Kimmo and Asko Parpola. 1980. Documentation and duplicates of the texts in the Indus script. Tech. rep., University of Helsikni. Department of Asian and African Studies, Research Reports 2.

Koskenniemi, Kimmo and Asko Parpola. 1982. Corpus of texts in the Indus script. Tech. rep., University of Helsinki. Department of Asian and African Studies, Research Reports 3.

Koskenniemi, Seppo, Asko Parpola, and Simo Parpola. 1970. A method to classify characters of unknown ancient scripts. *Linguistics* 61:65–91.

Magerman, David and Mitchell Marcus. 1990. Parsing a natural language using mutual-information statistics. In *Proceedings of Eighth Annual Meeting of AAAI*.

Mohri, Mehryar and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. In *Association for Computational Linguistics, 34th Annual meeting*, pages 231–238. ACL, Santa Cruz, CA.

Parpola, Asko. 1994. *Deciphering the Indus Script*. Cambridge: Cambridge University Press.

Prince, Alan and Paul Smolensky. 1993. Optimality theory. Tech. Rep. 2, Rutgers University, Piscataway, NJ.

Roark, Brian and Richard Sproat. 2006. *Computational Approaches to Syntax and Morphology*. Oxford: Oxford University Press. Forthcoming.

Shannon, Claude. 1948. A mathematical theory of communication. *Bell System Technical Journal* 27:379–423, 623–656.

Sproat, Richard and Chilin Shih. 1990. A statistical method for finding word boundaries in Chinese text. *Computer Processing of Chinese and Oriental Languages* 4:336–351.

Sproat, Richard and Jan van Santen. 1998. Automatic ambiguity detection. In *Proceedings of the International Conference on Spoken Language Processing*. Sydney.

14

Consonant Gradation in Estonian and Sámi: Two-Level Solution

TROND TROSTERUD AND HELI UIBO

14.1 Introduction

Koskenniemi's two-level morphology was the first practical general model in the history of computational linguistics for the analysis of morphologically complex languages. In this article we will reconsider one of the key innovations in Koskenniemi (1983), namely the treatment of consonant gradation in finite state transducers. We will look not at Finnish, but at two languages with a more extensive consonant gradation system, namely Estonian and Sámi. The goal of the paper is to demonstrate two different ways of modeling consonant gradation in a finite state morphological system - lexical and morphophonological. We will also compare the resulting systems by their computational complexity and human-readability.

Consonant gradation is rare among the languages of the world, but stem alternation in itself is not, and the treatment of consonant gradation can readily be transferred to other stem alternation phenomena. Koskenniemi's original idea was to see stem alternation as an agglutinative phenomenon. Consider the example (14.1), showing a two-level representation of stem alternation.

$$ehTe\$: ehe \tag{14.1}$$

Here the $ sign is a quasi-suffix, introduced to trigger consonant gradation in the stem. Two-level rules decide the correspondence of T to surface phonemes t or 0 (empty symbol), based on the context, specifically, according to the presence or absence of the symbol $ in the right context.

Another type of rules for handling stem alternations that can be compiled

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

into finite state automata is the method of sequentially ordered replace rules presented in Karttunen (1994) which have the format shown in (14.2):

$$a -> \ b \,||\, LC \ \ RC \tag{14.2}$$

The rule should be interpreted like "*a* is replaced by *b* in the left context *LC* and right context *RC*". The main practical preference of the replace rules compared to two-level rules is that they can handle a segment consisting of several characters as a whole, whereas handling the change of a character string by two-level rules requires several rules to be co-ordinated, one for each character alternation. This is, for instance, useful for building the additive forms for some inflection types in Estonian where the lemma has duration I, but the singular additive form has duration III. This case is especially difficult for stops, the III grade of which is not built by simple doubling but the double consonant is different from the corresponding I grade phoneme, cf. the nominative and short illative (additive) forms *rida:ritta* 'row', *tuba:tuppa* 'room', *nägu:näkku* 'face'.

This kind of change can be handled by **one** replace rule (14.3) but requires **two** two-level rules. In two-level rule system it also requires the introduction of new lexical symbols (=, 2) which invoke the rules in (Figure 14.1).

$$d -> \ tt \,||\, V \_ V2 :; \tag{14.3}$$

```
WeakStop:Stop <=> _ %=: (StemVowel:) 2:;
     where WeakStop in (g G b B d D)
                Stop in (k k p p t t)
     matched;

%=:Stop <=> WeakStop: _ (StemVowel:) 2:;
     where WeakStop in (g G b B d D)
                Stop in (k k p p t t)
     matched;
```

FIGURE 1 Handling grade alternation I-III by two-level rules

On the other hand, the strong preference of two-level rules is that there should not been defined any ordering on them as they work independently from each other. We have made some experiments with replace rules in the early stage of building Estonian finite state morphology. Based on our experience it was quite difficult to write a consistent replace rule set, as some higher-priority rules often spoiled the contexts for some of the lower-priority rules. Writing a **consistent two-level rule set** turned out to be considerably easier.

These advantages / disadvantages have the same source. As the empty symbols are coded as real zeros in the lexical level of the two-level model, the two-level rules (as finite state transducers) can be intersected. And this approach does not allow "unequal" changes like *b -> pp*, where one character is replaced by two. Replace rules can replace a segment of arbitrary length by another segment of arbitrary length. On the other hand, replace rules cannot be intersected, but should be applied sequentially instead.

More or less the same advantages have also been pointed to in Karttunen (2001):

> From the current point of view, two-level rules have many interesting properties. They are symbol-to-symbol constraints, not string-to-string relations like general rewrite rules. Two-level rules make it possible to directly constrain deletion and epenthesis sites because the zero is an ordinary symbol. Two-level rules enable the linguist to refer to the input and the output context in the same constraint.

From a formal point of view there is no substantive difference: a cascade of rewrite rules and a set of parallel two-level constraints are just two different ways to decompose a complex regular relation into a set of simpler relations that are easier to understand and manipulate Karttunen (2001). Thus, it is more like matter of taste, i.e. what kind of rule system seems better to grasp for the individual language engineers. We have opted for two-level rules but that does not mean we exclude the possibility of using replace rules at all. At the moment only two-level rules are used in the description of Estonian morphology but it is possible to code *b -> pp*, *d -> tt* etc. rules as replace rules instead.

In their historical overview of the development of finite state transducers, Karttunen (2001) pointed out that one problem with two-level transducers in the early years was connected to hardware limitations:

> It was also known from the beginning that a set of two-level transducers could be merged into a single one (...) by intersecting them. The problem is that in both cases the resulting single transducer is typically huge compared to the sizes of the original rule networks. Composition and intersection are exponential in the worst case. That is, the number of states in the resulting network may be the product of the sizes of the operands. Although the worst case is purely theoretical, in practice it turned out that intersecting large two-level rule systems was either impossible or impractical on the computers available in the early 90s.

In the present article we would like to look at the efficiency issue of two-level rule systems again, in the light of the processor speed of contemporary computers.

14.2 Method

In addition to giving a descriptive overview of the finite state systems of Estonian Northern and Lule Sámi, we are also going to give some characteristic numbers in order to measure the rule sets and lexicons of each system, thereby givingsome ground for comparison.

We will compare the morphological transducers of Northern Sámi, Lule Sámi and Estonian. Linguistically, we may say that Estonian and Sámi are similar as regard to the number of stem variants for the words with consonant gradation - usually two, but in some cases there are even four. While comparing we have to bear in mind that the systems are in different stage of development, as regard to their lexical coverage.

In section 14.5.1, we will use the following units of measure:

- number of records per lexical unit in stem lexicon;
- number of continuation lexica per lexical unit;
- number of states and arcs in the resulting morphological transducer (which is the composition of lexical and rule transducers);
- time of compilation of the rule (and lexicon) transducer.

14.3 Consonant gradation types in Estonian and Sámi

14.3.1 Consonant gradation types in Estonian

There are three different phoneme durations in Estonian (I - short, II - long and III - extra long). In written form the durations II and III are identical (written as a double vowel/consonant or a cluster of 2-5 consonants or two vowels), except for the stops where there are three different written forms as well (I - *g, b, d,* II - *k, p, t,* III - *kk, pp, tt*). There are two principally different consonant gradation types in Estonian - qualitative and quantitative.

1) Qualitative changes

1a) deletion of a stop (g, b, d, k, t) or s (table 1).

1b) assimilation (*kandma : kannan* 'to carry', *vars : varre* 'stalk');

1c) replacement of a weak stop by rules b:v, d:j, g:j (*kaebama : kaevata* 'to complain', *rada : raja* 'path', *märg : märja* 'wet');

arg	:	ara	fearful	käskida	:	käsin	to order
tuba	:	toa	room	ehte	:	ehe	adornment
vedama	:	vean	to transport	mesi	:	mee	honey
uskuda	:	usun	to believe				

TABLE 1 Deletion of g, b, d, k, t, s

Additionally, in some inflection types with the gradation type 1a) the singular additive form is in duration III (cf. the examples presented in section

14.1). The same occurs in another inflection type where in all other cases the stem remains unchanged (sg nom *pesa* 'nest', sg gen *pesa*, sg part *pesa*, sg addit *pessa*).

2) Quantitative changes

2a) alternation of long and short geminate (table 2, I column);

2b) alternation of strong and weak stops (table 2, II column);

kk : k	pikk : pika	long	k : g	vilkuda : vilgub	twinkle
pp : p	sepp : sepa	smith	p : b	kubjas : kupja	taskmaster
tt : t	võtta : võtan	to take	t : d	kartma : kardan	to be afraid
ss : s	kirss : kirsi	cherry			

TABLE 2 Estonian quantitative gradation

Estonian differs from Finnish, where consonant gradation is a weakening process only, in also having some noun inflection types with strengthening quantitative consonant gradation, although the weakening consonant gradation is considered the main type of consonant gradation.

Weakening consonant gradation is defined as follows:

- nouns: sg nom (sg part) - strong grade, sg gen - weak grade
- verbs: supine (primary form) - strong grade, indicative mode present tense - weak grade

In the paradigms of words with strengthening consonant gradation the strong and weak grade stems occur just conversely.

The strengthening consonant gradation types of nouns are the following:

a) nouns that derived from a verb with consonant gradation, e.g.: *hinne : hinde* 'mark' (verb *hindama - hinnata - hindan* 'to evaluate')

b) nouns that end with s and are in weak grade in singular nominative, but singular genitive is in strong grade and the final s is deleted, e.g , *saabas : saapa* 'boot'.

c) nouns that end with vowel + r (*vaher : vahtra* 'maple', *tütar : tütre* 'daughter')

d) nouns that additionally to the gradating stem have stem final change e-me (*liige : liikme* 'member', *võti : võtme* 'key')

There are no verb inflection types with strengthening consonant gradation in Estonian.

14.3.2 Consonant gradation in Sámi

In essence, Sámi consonant gradation is a phenomenon quite similar to its Finnish and Estonian counterpart. The consonant cluster on the border of the final and antepenultimate syllables of the stem, may change, i.e. the consonant

cluster has different grades. Typically, there are two grades, strong grade and weak grade. Sámi and Estonian are among the few gradation systems with three grades, but in most cases the grade alternation is binary, i.e. III-II or II-I. Thus, a grade II consonant cluster may be strong relative to a grade I cluster, and weak relatively to a grade III cluster.

Historically speaking, strong grade was found in the consonant at the onset of the final or penultimate open syllable in a stem, whereas the grade changed to weak when inflectional processes closed the final open syllable (and vice versa, for consonant-final stems). In the modern languages, the triggering environment for consonant gradation is (inflectional or derivational) morphology.

We will look at the consonant gradation pattern of Lule Sámi, and in some special cases also at the pattern found in Northern Sámi. Linguistically speaking, they have the same consonant gradation system, but this pattern is represented in different ways in the respective orthographies of the two languages. Since the automata presented in this article generalise over written language, rather than over phonological representations, Lule and Northern Sámi consonant gradation must be treated as being more different than they are in the spoken language.

One difference is that Lule Sámi makes more use of digraphs, i.e., instead of *š* they write *sj*. In the two-level morphology formalism, each alternating symbol must get its own rule (many-to-many alternation is not allowed), this calls for more rules than the Northern Sámi gradation. On the other hand, in Lule Sámi going from strong to weak grade is a uniform process, letters are either changed or deleted, whereas in Northern Sámi letters may be either changed, deleted or added. Compare the following parallel forms, where the strong-weak alternation is denoted as $xy : xyy$ in Northern Sámi, and as $xyy : xy$ in Lule Sámi. Linguistically speaking, the gradation is in both cases of the same type *xøy:xy*, where *ø* = schwa.

‘stone’	nominative	genitive
Northern Sami	geađgi	geađggi
Lule Sámi	giergge	gierge

TABLE 3 Northern Sámi $xy : xyy$ and Lule Sámi $xyy : xy$

We will first look at quantitative gradation, and then at qualitative gradation. Finally, we will look at a mixed type.

Quantitative alternation involves fricatives, liquids and voiced nasals, 34 alternating pairs in Northern Sámi and 57 alternating pairs in Lule Sámi. In one subtype of the qualitative gradation, grade I is written with single consonant, and grade II with double consonant. In the standard orthography, grade

III is also written with double consonant. In earlier orthography, grade III was written with an apostrophe, and in order to give a linguistically adequate representation, our transducer also accounts for III-II gradation, although it is not visible in output mode. In future applications involving text-to-speech, the orthographically invisible III-II alternation will become relevant, as we via disambiguation will be able to predict the correct grade of nominative (grade III) and accusative/genitive (grade II) nouns, although they are written identically. An example from Northern Sámi is *oađ'đi : oađđit : oađán* 'sleeper : to sleep : I sleep'.

A single, nongeminate letter may also be deleted. There are no examples in Sámi of an intervocalic consonant being deleted (of the Estonian type $tuba : toa$, but consonants that are part of consonant clusters may be deleted, as in the Lule Sámi pairs *jiegńa : jieńa* 'ice Nom:Gen', *spádnjo : spánjo* 'birch forest Nom:Gen'. Due to orthographical convention, one qualitative alternation in Lule Sámi is written as if it were a quantitative one, *htj:tj, hts:ts*, cf. *biehtse : bietse*, 'spruce Nom:Gen'. The other alternation belonging to this type are treated in section 14.4.2 below.

There are several types of qualitative consonant gradation. The simplest case is found in Lule Sámi, where one letter is changed into another one, like in *oakse : oavse* 'branch Nom:Gen' and *bákte : bávte* 'cliff Nom:Gen'. Only k undergoes this change, in consonant clusters with *s, t, tj* and *ts*. One-consonant changes are found in Northern Sámi as well, but with more complex context, as for the *rbm:rpm, rdn:rtn, rg*η*:rk*η, pairs, e.g. *fierbmi : fierpmi* 'net (Nom:Gen)'.

A different type of qualitative alternation is the $xx : yy$ type voice alternation where voiced stops and affricates change into unvoiced stops and affricates. The pairs are *bb/pp, dd/tt, gg/kk*, in Northern Sámi also *ddj/dj, zz/cc, žž/čč*, cf. Lule Sámi *oabbá : oappá* 'sister Nom : Gen'. Lule Sámi has the three latter alternations, but due to the different orthographical principles, they are written as *dtj:ttj, dts:tts*, and pattern with the *ks:vs* alternation, as far as two-level rules are concerned.

One type of qualitative consonant gradation is preaspirated stops and affricates change into their voiced counterparts. The II-I pairs in Northern Sámi are *hp:b, ht:đ, hk:g, hdj:j, hc:z, hč:ž*, the Lule Sámi ones are *hp:b, ht:d, hk:g*. In grade III, the stops from grade II are doubled. Cf. the Northern Sámi series *ohcci : ohcat : ozan*, 'searcher : to search : I search', a three-grade inflection pattern. A corresponding example for Lule Sámi would be *jåhtte : jåhtet : jådåv* 'mover : to move : I move'.

14.3.3 Similarities and differences between Sámi and Estonian

We see that Sámi consonant gradation is more complex and variable than Estonian consonant gradation but there also exist a common part. Compared to

the more well-known Finnish gradation, Sámi and Estonian are both more complex: More letters are involved in the alternation, a larger part of the lexicon is affected by the alternation. Contrary to Finnish, both Sámi and Estonian have a 3-way opposition, where the strongest grade III in certain cases is invisible in writing. The bulk of the alternation involves a binary opposition III/II or II/I, but there are also cases of III/II/I alternation within a single paradigm.

Typologically speaking, it is no accident that Estonian and Sámi differ from Finnish in another respect as well: Due to several apocopy processes, the segmental morphology has been shortened, and in many cases even disappeared (as in the important Genitive case). This has given consonant gradation a more prominent position in the grammar of Estonian and Sámi. Seen from a computational point of view, this typological difference is of no importance. In all three languages, consonant gradation is a non-segmental morphological operation, which must be triggered by elements introduced via the morphological process.

From the computational point of view, a more important difference is the non-existence of the stem final vowel in the lemmas of some noun types of Estonian: the stem vowel appears in inflected (genitive) stem but not in lemma stem. For handling this phenomenon, the two-level model provides us with a sufficient toolset. It is possible either to include the stem final vowel to into the lexical representation of the stem and force it to be deleted for singular nominative. And it is also possible to add the stem vowel to the inflected stem in a continuation lexicon. The morphological description of Estonian uses the second approach.

In the following section we will see how the consonant gradation processes have been described by the means of finite state morphology. The research have been done independently, thus coming to the similar solutions is incidental. And in some cases we have used different means to describe similar processes.

14.4 The finite state description of Estonian and Sámi morphology

14.4.1 Two-level morphology of Estonian

The morphological description of Estonian has been built up, lead by the principles of two-level morphology model (Koskenniemi (1983)). It consists of a network of lexicons and a set of two-level rules.

The two-levelness of the model means that the lexical representations of morphemes are maintained in the lexicons and the task of two-level rules is to "translate" the lexical forms into the surface forms and vice versa. The lexical forms may contain information about the phoneme alternations, about

the structure of the word form (morpheme and compound boundaries) etc.

The most optimized system of inflection types of Estonian Viks (1992) includes 38 types - 26 noun types and 12 verb types. 14 noun types and 10 verb types are the types which have some kind of consonant gradation (including the types where the gradation is not visible in the written form). As the system deals with written language, only the inflection types with qualitative changes in stem and types with quantitative stop gradation are of our interest. There are 15 such inflection types according to Viks (1992) in Estonian – 10 noun types and 5 verb types.

The main principle in describing Estonian consonant gradation has been to keep the lexical representation as readable and meaningful as possible. We have used the capital letters K, P, T, G, B, D, S to mark the phonemes which undergo some kind of change (deletion, assimilation) in the inflection processes. Additionally, the character $ is used to mark the weak grade (similarly to Koskenniemi (1983)).

```
 Lexicon Nimisona                                  Lexicon 18
hamBa   07_S-0;   poisS   23_I;                       TP_18at;
jalG    22_A;     riD=a   18_Adt_PlPV;            :$ TP_18an;
jıG=i   18_Adt;   siGa    18_PlPV;
laD=u   18_Adt;   sıD=a   18_Adt_PlPV;
laG=i   18_Adt;   teGu    18;
luG=u   18_Adt;   tiGu    18;
maDu    18;       tikK    22_U;
maG=u   18_Adt;   tekK    22_I;
manDEr  03_I;     tuB=a   18_Adt_PlPV;
paTj    24;       vahTEr  03_A;
```

TABLE 4 Presentation of the stems with consonant gradation in the root lexicon

Note that there is only lexical representation given in the root lexicon, not as it is usually done in lexical transducers (e.g. $tigu + S : tiGu18$) where lemma and morphological information are given on the left side of the transducer and lexical representation of the word-form is on the right side. The considerations for this kind of solutions are discussed in Uibo (2005). The capital vowels are subject to deletion synchronously with grade alternation. The symbol = is used to mark the consonants that are subject to gemination when building singular additive (corresponding rules given in section 14.3.1). The next level of lexicons (Lexicon 18 in Table 4) divides the word-forms between strong and weak grade referring to the corresponding continuation lexicons TP_18at and TP_18an. For the weak grade stems the $ sign is added at the end of the stem.

The two-level rules are convenient to handle phoneme alternations, concerning only one phoneme. If the stem change is more complex (e.g. *idu:eo*), then it can be handled analytically.

Let us consider an inflection type in Estonian, which is characterized by weakening stem inflection (the deletion of phoneme *b*, *d*, *g* or *s*) and also changes in the immediate neighborhood of the disappeared consonant - the lowering of the surrounding vowels.

Example list of words belonging to the type is given in Table 5.

madu	:	mao	snake	lugu	:	loo	story
siga	:	sea	pig	käsi	:	käe	hand
pidu	:	peo	party	nuga	:	noa	knife
tegu	:	teo	action	süsi	:	söe	coal
uba	:	oa	bean				

TABLE 5 Weakening stem inflection in Estonian

A rule for handling the deletion in Table 5 is found in Figure 2:

```
"b,d,g,s deletion"   ($ marks the weak grade)
  LV:0 <=> Vowel: _ Vowel: $:;
```

FIGURE 2 "b,d,g,s deletion"

The immediate right and left contexts of the deletion rule (figure 2) are identical (`Vowel:`), they refer to any underlying vowel. The rule for vowel lowering (figure 3) has two distinct contexts: the vowel lowering may occur before (*siga : sea*) or after (*madu : mao*) the consonant gradation, in the first case the consonant gradation is part of the right context, and in the latter case it is part of the left context.

```
HVow:LVow <=> Bgn _ LV: StemVow: %$: ;
              Bgn Vow: LV: _ %$: ;
              where HVow in (u ü i)
                    LVow in (o õ e)
                    matched ;
```

FIGURE 3 "Vowel lowering"

In the paper Uibo (2000) the stem flexion types and the discovery process of rules have been discussed in details. The most problematic morphophoneme in Estonian is *D* which may correspond to five different surface

phonemes in weak grade: $D : 0$, $D : l$, $D : n$, $D : r$ and $D : j$. There the only way was to differentiate the correspondences by very detailed context. And luckily, the contexts do not overlap.

The number of consonant gradation rules in Estonian two-level morphology is 16 - this is the number of different lexical-surface character pairs that correspond to the weak grade (strong grade is considered default and weak grade - marked).

14.4.2 Finite state morphology of Sámi

Sámi consonant gradation is intertwined with many other morphophonological processes, such as stem vowel alternation and diphthong simplification. We use dummy elements to trigger the different morphophonological processes.

We will present two approaches to representing the consonant gradation types with two-level automata, the process-wise and the segment-wise approach, respectively. We will concentrate upon three types of alternations: The quantitative alternation (*ff:f*), the voicing alternation (*bb:pp*) and the seemingly inverted alternation *ig:igg* for the Northern Sámi schwa alternation.

14.4.3 Process-wise vs. segment-wise alternation

In the two-level formalism, we may generalise over either consonant gradation type (i.e., over context) or over alternating letter. We illustrate both options with an example from Lule Sámi. First we give a rule for the Lule Sámi consonant alternation $rgg : rg$, the rule in 4 (as a rule collapsing the 19 different consonant gradation patterns of this type that can be found in Lule Sámi). We may note that *g* takes part in another consonant gradation pattern as well, in the *g:ń* pattern in Figure 5, with 3 other consonant pairs.

```
Cx:0 <=> Vow: Cx _ Cy Vow ( StemCns: ) WeG: ;
where Cx in ( b d d g k l l l m m n p p s s s s ń ń )
      Cy in ( m j n ń n d j t b p d s t k m n t g k )
      matched ;
```

FIGURE 4 "Gradation Series 1, III-II, three-letter patterns"

```
Cx:0 <=> Vow: _ Cy Vow ( StemCns: ) WeG: ;
where Cx in ( b d d g )
      Cy in ( m j n ń )
      matched ;
```

FIGURE 5 "Gradation Series 1, II-I, two-letter patterns"

Alternatively, one may choose to analyse the alternation in question not as a generalisation over multiple types, but as a generation over multiple contexts, i.e. write one rule for each of the 10 consonant that are involved in the 23 alternation patterns of the 2 rules above. The result, for g, may be seen in Figure6:

```
g:0 <=> Vow: ( g ) _ ń Vow: (StemCns: ) WeG: ,
        Vow: [ j|l|r|v ] _ g Vow: ( StemCns: ) WeG: ;
```

FIGURE 6 "Consonant gradation g:0"

The Lule Sámi consonant gradation was analysed in both ways. Ordered according to context, the set contains 20 rules, ordered according to alternating consonant, it contains 25 rules. When ordered according to alternating consonant, each rule contains appr 4 subrules, thus the total number of rules in the latter approach is 74.

The computational difference between the two is that ordered according to context, the rule set contains a large number of conflicting contexts, who must be resolved by the parser. The parser is good at it, but it takes time, a quarter of an hour on a not too fast machine, as a matter of fact. Comparing the compilation time between the two rule sets (on a 400 MHz Power Mac G4), we se a huge difference, cf. Table 6.

Rule set ordered according to:	# of rules	# of subrules	Compilation time real	user	system
alternating consonant	4	16	0m11.013s	0m1.140s	0m0.240s
context	4	4	16m14.387s	3m8.250s	0m7.430s

TABLE 6 Compilation time

We discuss the compilation issue at the end of paragraph 14.5.1.

Schwa alternation

This alternation was represented in Table 3 above. In Lule Sámi, this alternation may be analysed in the same way as the quantitative alternation type $xyy : xy$ found in pairs like *liehppa:liehpa* 'shelter Nom:Gen'. The phonological realisation is different in the two cases, but from a computational point of view, this is irrelevant. In Northern Sámi, the gradation in question is written $xy : xyy$, and here this alternation must be analysed in a different way. The method chosen was to represent the strong grade underlyingly as $x'y$, and to replace the apostrophe with the consonant to the right in the weak grade, and then to prevent any apostrophe from the surface representation.

14.5 Comparing the treatments

14.5.1 Consonant gradation as lexeme property or as lexicon property

Originally, consonant gradation was a phonological alternation, which affected phonologically defined consonant clusters in phonologically defined environments. As we have seen, the environments have now become morphological, and must be treated as such. When it comes to the gradating consonants themselves, the situation is not that clear. Most of the phonologically appropriate stems undergo consonant gradation, but not all of them do. The gradation types are not equally regular, in Estonian, for example, the qualitative gradation forms a closed class, whereas the some types of the quantitative consonant gradation are regular and productive.

In principle, there are two ways of dealing with this:

1. Alternating and non-alternating consonant clusters are not distinguished in the lexicon, rather, they are directed to different sublexica, and treated differently there.
2. Alternating and non-alternating consonant clusters are pointed towards the same sublexica, hence they have the same morphology. The difference is found in the stem, where the consonant clusters are given different archiphonemes. Either the alternating or the non-alternating consonant may be given the special phoneme.

The Sámi words *goahti* 'hut' and *stáhta* 'state' both contain the consonant cluster -*ht*-. The former alternates with -*đ*-, and the latter does not. This difference may be handled in two ways, denoted a and b in Figure 7.

```
a. Directing gradating and non-gradating to different lexica
    LEXICON NounStems
    goahti GRADATING-BISYLL-NOUN  ;
    stáhta NONGRADATING-BISYLL-NOUN  ;

b.i One continuation lexica, but marking the gradating noun
    LEXICON NounStems
    goahti:goahTi  BISYLL-NOUN  ;
    stáhta  BISYLL-NOUN  ;

b.ii One continuation lexica, but marking the non-gradating noun
    LEXICON NounStems
    goahti  BISYLL-NOUN  ;
    stáhta:stáhTa  BISYLL-NOUN  ;
```

FIGURE 7 Two strategies for continuation lexica

In a., the subsequent lexicon GRADATING-BISYLL-NOUN would contain a consonant gradation trigger not present in NONGRADATING-BISYLL-NOUN. In b.i, we would have a morphophonological rule changing

T to đ, and another rule deleting h in front of a T:đpair. Solution b.ii would be the mirror image of b.i, having gradation as the default case, with a rule deleting all instances of t (but not T) in the context h_V, for the relevant word-forms, and with a rule rewriting T as t in all contexts.

Whether to choose b.i or b.ii is a matter of taste. If consonant gradation is the rule and not the exception, it is of course tempting to treat the exceptions as such. On the other hand side, giving gradation the special treatment makes it easier to control: Gradation occurs where we have said that it should, and nowhere else. Both Koskenniemi (1983) and the present treatment of Estonian thus chose the b.i option.

The Sámi solutions presented here opt for alternative a. This gives a simpler stem lexicon but more complicated continuation lexica. And indeed, the Northern Sámi transducer has 250 continuation lexica for the noun, adjective and adverb complex, as compared to the somewhat lower 164 for Estonian. Note that the number of continuation lexica is also dependent upon the coverage of the transducer. The Lule Sámi transducer has a weaker coverage for derivational processes, and a somewhat more regular adjective declension pattern, and here the number of continuation lexica is 108.

For our Southern Sámi transducer we have chosen option b.i, and although the numbers cannot be compared directly (Southern Sámi does not have consonant gradation, but its Umlaut phenomenon is of compatible size and complexity), it contains only 29 continuation lexica.

	records per	root				Processor	Compilation time	
language	lex unit	lexicon	states	archs	paths	MHz	rule trans	lex trans
Estonian	109/55=1.98	400	1,940	5,009	circular	700	3s	0s
Lule S.	166/48=3.45	760	2,413	4,722	755,374	1,400	0.5s	1.6s
North. S.		75,294	95,652	258,350	circular	1,400	1m 6.2s	2m 38.9s
Lule S.						400	6.2s	12.2s
North S						400	5m 6.5s	7m 25.6s

TABLE 7 Comparing the compilation of Estonian and Sámi

In evaluating the results shown in Table 7, one has to be aware that the rule sets have been built based on different principles - Northern Sámi uses the context-oriented approach, whereas in the rule sets for Estonian and Lule Sámi each rule handles a concrete pair and lists all the possible contexts disjunctively on the right side of the rule. There are lots of formal conflicts in the Northern Sámi rule set, which is reflected in the compilation time. As long as compilation time is not a critical factor, the context-oriented approach of Northern Sámi is fine, but writing two-level rules relative to the alternations will reduce compilation time drastically.

14.6 Conclusion

The two-level morphology compiler *twolc* is fully capable of handling even large and complex grammars. However, in a longer perspective we could try to combine two-level and replace rules (another tool from the Xerox finite state package – *xfst* – can be used for that purpose), as some kind of rules are more convenient to be handled by replace rules.

Non-segmental morphology may be handled by abstract, segmental rule triggers.

If compilation time is a factor, then context conflicts should be resolved before compilation. Still, even for large rule systems compiled on machines as slow as 400 MHz, this only gives 7 min as compilation time. During a developmental phase this may be a nuisance, but it can be lived with. And better source code reduces the compilation time to seconds.

We have shown that the finite state system of lexicons and rules both of which are computationally finite state transducers is very flexible: the system builder can choose if (s)he wants to describe a certain phenomenon by rules or by lexicons. As a rule of thumb, stem changes are more likely to be described by rules and morpheme combination rules by lexicons, but as we have seen, some types of the stem changes can be more naturally described by continuation lexicons.

References

Karttunen, Lauri. 1994. Constructing lexical transducers. In *15th International Conference on Computational Linguistics (COLING-94)*, pages 406–411. Kyoto, Japan.

Karttunen, Lauri. 2001. A short history of two-level morphology. *Sámi diedalaš áigečála* 3:100–123.

Koskenniemi, Kimmo. 1983. *Two-level Morphology: A General Computational Model for Word-form Production and Generation*. Publications of the Department of General Linguistics, University of Helsinki. Helsinki: University of Helsinki.

Uibo, Heli. 2000. Kahetasemeline morfoloogiamudel eesti keele arvutimorfoloogia alusena. In *Tartu Ülikooli Üldkeeleteaduse Õppetooli toimetised 1: Arvutuslingvistikalt inimesele*, pages 37–72.

Uibo, Heli. 2005. Optimizing the finite-state description of Estonian morphology. In *15th Nordic Conference on Computational Linguistics, NoDaLiDa 2005*.

Viks, Ülle. 1992. *A concise morphological dictionary of Estonian*. Tallinn: Eesti keele instituut.

15

RUSTWOL: A Tool for Automatic Russian Word Form Recognition

LIISA VILKKI

15.1 Introduction

The purpose of this paper is to describe the earlier version of RUSTWOL as a tool for automatic Russian word form recognition. The theoretical foundation of the RUSTWOL program is the two-level model, a language-independent model of morphological analysis and synthesis by Kimmo Koskenniemi (1983). My description is based on a document written by me, when I was working as a linguist at Lingsoft (Vilkki 1997). This earlier version of RUSTWOL was later used at Lingsoft as a basis for a new format. The newer version of RUSTWOL, representing the new format, and its documentation, written by me, are currently available at Lingsoft for customers only.

The main motivation for turning back to history and describing the first quasi-final version of RUSTWOL is that RUSTWOL is one of the most large-scale morphological programs in the tradition of the two-level formalism - yet my document at Lingsoft (Vilkki 1997) is the only presentation of it. The second motivation is that this earlier version has been useful at the Department of Slavonic and Baltic Languages and Literatures at University of Helsinki for the purposes of research and teaching. In 2000, I used this version of RUSTWOL for the morphological analysis of the Russian Corpus of Newspaper Articles, which is available in the University of Helsinki Language Corpus Server (UHLCS) and, in addition, at the University of Tampere. Recently, some changes in the RUSTWOL lexicon and rules have been made by Alexander Paile (2003) for the purposes of the HANCO project (Kopotev, Mustajoki 2003).

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

In Koskenniemi's (e.g. 1983,1997) two-level formalism, morphological phenomena are described as relations between lexical and surface levels. One character of the lexical level corresponds to one or a null character of the surface level. Therefore, the two-level formalism is neutral with respect to analysis and generation of word forms.

The RUSTWOL alphabet includes letters, numbers, special characters and diacritic symbols. The surface representations of the word forms are translations of their Cyrillic ortography.

The main components of TWOL are the lexicon and the two-level rules. The RUSTWOL lexicon defines lexical entries and their representations. It also specifies the morphotactic structure of the language and includes part of the morphophonological alternations. The RUSTWOL rule component contains 50 rules, and it deals with fairly natural, transparent alternations. There are also rules for the correct combination of stems and endings. These rules refer to the diacritic symbols of masculinity, animacy, transitivity, reflexivity and imperfective or perfective aspect. In addition, some of the rules control the correct combination of the parts of the compound words. Because the length of this contribution is restricted, it is not possible to describe here the rule component in detail. In addition, only some parts of the lexicon can be focused on.

15.2 An overview of RUSTWOL

The basic lexicon of RUSTWOL is based on a machine-readable version of Zaliznjak (1987). The complete material of this dictionary was not included in the lexicon of RUSTWOL. The dictionaries Jevgen'eva (1981-1984) and Zasorina (1977) were used in order to exclude, e.g. very infrequent words, some words representing special vocabulary and words marked stylistically as colloquial, archaic or local. The word material of the Zaliznjak lexicon was completed with words from Scheitz (1986), Kotelova (1984) and Kahla (1982,1984).

The most productive derivated forms and compounds, listed in Zaliznjak (1987) in their own entries, are treated in RUSTWOL by continuation classes and by the mechanism of compounding. Apart from Zaliznjak, Švedova (1982), Kuznecova and Efremova (1986), Tihonov (1985) and Bukčina and Kalakučkaja (1987) were used on matters relating to the description of derivational morphology and compounding. The evolving RUSTWOL was tested on various text corpora: newspaper and magazine articles and literary texts. These corpora are included in Helsinki Corpus of Russian Texts, which are available in UHLCS.

RUSTWOL described in this paper has a lexicon of approximately 72,000 words. This number is considerably increased by a derivational morphology

and the mechanism for compounding.

RUSTWOL assigns the possible readings of Russian word forms. The readings consist of the base form of the word and the morphological information of the inflected form.

Word form

Reading: "base form" morphological analysis
Reading: "base form" morphological analysis
etc.

RUSTWOL is meant to be used as the basis morphological tool in, for example, text analysis, spelling correction and information retrieval. Here are some central aims and properties of RUSTWOL:

- analyses written standard Russian
- gives the morphological information of Russian word forms
- has a complete inflectional morphology and a fairly extensive derivational and compounding morphology
- contains the basic vocabulary of Russian
- prefers traditional morphological categories

15.3 Lexicon

The lexicon of RUSTWOL consists of sublexicons connected to each other. The sublexicons include mostly root entries but also some full-form entries. Examples of root entries are given in 15.3.1. A full-form entry contains the whole word form and its morphological information. Only the extremely irregular word forms and some compounds, both parts of which are inflected (see 15.3.3), are coded as full-form entries. For example, a pronoun form **c1to-to** ‘something’ has the following two full-form entries:

c1to-to # “c1to-to INDEF PRON ACC”;
c1to-to # “c1to-to INDEF PRON NOM”;

15.3.1 Inflection

RUSTWOL incorporates a full description of Russian inflectional morphology. It uses 14 parts of speech. The parts of speech and other morphological properties of word forms are indicated by tags. To each word, a base form and at least one tag is associated.

Verbs

Verbs are labelled with V and they are identified by aspect, mood, tense, person and number, voice and reflexivity. Past tense forms are not identified by person and number but by gender or plurality. Here are some examples of the forms of a verb **delat’** ‘to do’:

delat'
"delat'" IMPF V INF ACT
delaet
"delat'" IMPF V PRES SG3 ACT
delal
"delat'" IMPF V PAST MA ACT

Participles are labelled with PART and verbal adverbs with V ADV. Long–form participles change according to gender, number and case. Short-form participles have a label SH. The following examples are forms of the verb **c1itat'** 'to read':

c1itaemyj
"c1itat'" IMPF V PRES PART MA SG NOM PASS
"c1itat'" IMPF V PRES PART MA SG ACC PASS
c1itaema
"c1itat'" IMPF V PRES PART PASS SH FE
c1itav
"c1itat'" IMPF V PAST V ADV ACT
c1itano
"c1itat'" IMPF V PAST PART PASS SH NE
"c1itat'" IMPF V PART PASS PRED

Verbs are divided into two conjugations (1V and 2V). Some verbs do not clearly belong to any of these conjugations (V). Nearly all inflectional types of verbs have an alternation pattern, which is a sublexicon that lists the lexical representations of suppletion-like alternatives. The stems of verbs are usually formed from roots and from some alternation pattern. Many inflectional types of verbs have variants for impersonal verbs. They are indicated by a tag IMPERS before other tags.

Correct combinations of stems and endings are defined by using proper continuation classes for each alternation entry or ending in the lexicons. In some cases rules are used for forbidding or permitting only certain combinations. For example, rules (33)-(39), concerning diacritics P~, P, V, V~, R and R~, exclude invalid combinations (see below).

Verbal endings are grouped into a few sublexicons. Continuation classes can also consist of a single minilexicon:

1V1: PRES PART ACT
1V2: PRES PART PASS, PRES V ADV
2V2: PRES PART ACT

For example, a verb **pet'** 'to sing' has the following entry that refers to the LEXICON oJ-e/1V:

```
pP~VR~ oJ-e/1V;

LEXICON oJ-e/1V
oJ 1V011 “et’”;
ojP~ 1V1 “et’Q4”;
oJP~ 1V2 “et’”;
e V021 “et’”;
eV V32 “et’Q5”;
```

The first stem, poJ, has the continuations 1V011, 1V1 and 1V2, and the second stem, pe, has the continuations V021 and V32. All verb entries of this inflectional type refer to this minilexicon. However, only the continuation classes in the first and in the fourth entry of the minilexicon are possible for all verbs of this inflectional type. In the second and third entry, P~ is a symbol of imperfective aspect. For all verb stems that are marked P, a symbol of perfective aspect, the continuation 1V1 and 1V2 are excluded.

There are three types of diacritics in verb inflection. They indicate aspect (P and P~), transitivity (V and V~) and reflexivity (R and R~). Most of the verb stems have either P or P~. In this way, perfective and imperfective stems of the same inflectional type can have the same ending minilexicons. Diacritic V is used in transitive verbs and V~ in intransitive verbs. Only stems marked V can get PAST PART PASS and PART PASS PRED ending. These endings are given in minilexicons V31 and V32. So before continuation classes referring to these minilexicons there is a diacritic V.

In minilexicon 1V2, the entry PRES PART PASS demands both V and P~. These diacritics are also necessary in the case of PASS REFL. Only stems having them can get PASS REFL ending -sa1 (in minilexicon RF0) or -s’ (in minilexicon RF1). These endings can be added after personal (PRES/FUT), past tense or infinitive endings. After PRES PART ACT or PAST PART ACT ending some adjectival ending is added and only after it reflexive ending.

All stems marked R get interpretation ACT REFL. These are called reflexive verbs. Many imperfective verb forms have both ACT REFL and PASS REFL interpretations. For example, a form **c1itau1s2ijsa1** of the verb **c1itat’** ‘to read’ is given the following interpretations:

```
c1itau1s2ijsa1
    “c1itat’” IMPF V PRES PART MA SG NOM ACT REFL
    “c1itat’” IMPF V PRES PART MA SG ACC ACT REFL
    “c1itat’” IMPF V PRES PART MA SG NOM PASS REFL
    “c1itat’” IMPF V PRES PART MA SG ACC PASS REFL
```

The first and the second interpretations represent the following entry:

c1itaP~V~R J-0/1V-R;

The third and the fourth interpretations are representations of a different kind of entry:

c1itaP~VR~ J-0/1V;

Some verb forms (V ADV, IMPV, PASS PART) cannot get PASS REFL ending, even if their stems have diacritics P~ and V. Therefore, these forms have continuations to minilexicons where PASS REFL interpretation is lacking.

Nouns

Nouns are given a tag N, and they are categorized by gender, number and case. A noun **dom** 'house' has, for example, these inflectional forms:

dom
    "dom" N MA SG NOM
    "dom" N MA SG ACC
dome
    "dom" N MA SG PREP

The main declension types of nouns are determined by gender: masculine (/1SM), feminine (/2SF and /3SF) and neuter (/1SN). All of them have subtypes. These are distinguished on the basis of, for example, **u/u1** ending in MA SG GEN and MA SG PREP, various exceptional plural forms and various alternation patterns.

Some of the subtypes are further divided into two types of minilexicons: words representing the first type cannot be used as the first parts of compound words, whereas words representing the second type can be used (see 15.3.3).

In addition, there are declension types and subtypes for words that are inflected like feminines but are used syntactically as masculines (/2SM) or either as masculines or as feminines (/2SMF) and for words that are inflected like neuters but are used syntactically as masculines (/1SMN). Words that cannot be inflected (/SM-ind, /SF-ind and /SN-ind) and words occurring only in plural have their own types, too.

A nominal declension type usually includes one or more continuation classes of singular forms and plural forms. Some continuation classes consist of only a single minilexicon.

Some endings in nominal ending lexicons have diacritics N (animate), N~ (inanimate) or M (masculine). Therefore, only noun stems having appropriate diacritics can get these endings. The combination of stems and endings is controlled by rules. For example, a noun **divo** 'miracle' has an entry of the following kind:

divM~N~ /1SN;

The stem **div** gets some of its endings in minilexicons 1SM1, SPL1 and SPL3. It has diacritics M~ and N~ and, therefore, SG ACC ending in minilexicon 1SM1 and PL ACC ending in minilexicon SPL3 cannot be added. By contrast, the stem can be combined with PL ACC ending in minilexicon SPL1. A diacritic Q3 in minilexicons is used for compound formation (see 15.3.3).

```
LEXICON 1SM1
    AQ3 TO " SG GEN";
    UQ3 TO " SG DAT";
    AMNQ3 TO " SG ACC";
    OmQ3 TO " SG INSTR";
    eQ3 TO " SG PREP";
LEXICON SPL1
    AQ3 TO " PL NOM";
    AN~Q3 TO " PL ACC";
LEXICON SPL3
    Q3 TO " PL GEN";
    NQ3 TO " PL ACC";
```

Other parts of speech

The description of other parts of speech is presented in Vilkki 1997. This version of RUSTWOL does not contain special labels for proper names and abbreviations. However, capital letters in these words have an asterisk (*). Proper nouns can be inflected in various declension types of adjectives, nouns and pronouns. Some of them, like all abbreviations, are not inflected.

15.3.2 Derivation

The version of RUSTWOL described here has only a system of first-degree derivation. Most of the adverbs and predicatives are derived from adjectives. Nouns with various suffixes are derived from adjectives or verbs.

Adverbial or predicative suffixes:

-o/e	otkryt-o
-i	a1nvarsk-i

Nominal suffixes:

-ost'/est'	a1dovit-ost'
-nost'	gotov-nost'
-stvo	grabitel'-stvo
-estvo	imus2-estvo
-instvo	dosto-instvo
(-jstvo)	bespoko-jstvo
-izm	biolog-izm
-'	glub-'
-ina	glub-ina
-nie	avla1-nie
-anie	z1ivopis-anie
-ovanie/evanie	absorbirov-anie
-a1nie	ble-a1nie
-enie	opolz-enie

Besides the kinds of derived words listed above there are, of course, many other kinds of derived words in Russian. The most frequent of these have entries in the lexicon.

15.3.3 Compounding

This first version of RUSTWOL has a mechanism for building compounds, mainly consisting of two parts. The most frequent compounds, consisting of more than two parts, are listed in the lexicon. Only the most productive first parts are chosen in productive compound formation. The bulk of the first parts can occur as independent words, too. The continuation classes of these roots include a continuation to the Stem1 or Stem2 lexicon as one alternative. Many compounds have a hyphen and/or a linking element **O** or **i** between the components. These are usually included in continuation classes. The linking element **O** is realized as **o** or **e**.

Word forms that are permitted as second parts have the following diacritics:

Q1+F1: qualitative adjective (long forms and short forms)
Q2+F1: relative adjective (long forms and short forms)
Q3: noun
Q4: present participle active
Q5: past participle passive

Most of the second parts can be used as independent words. Compounds that are listed in the lexicon have the diacritics mentioned above, too. In this way, the mechanism also permits compounds, consisting of more than two parts. The first parts have a diacritic C1, C2, C3, C4, C5, B1, B2 or B3. They permit the second parts of the following types:

C1: Q2+F1

C2: Q1+F1
C3: Q2+F1, Q3
C4: Q2+F1, Q3, Q4
C5: Q1+F1, Q2+F1
B1: Q3
B2: Q4
B3: Q5

The correct combination of the parts is controlled by rules (40)-(48).

In order to treat compounds, the vocabulary is split up into four main lexicons. Stem1 is the largest one. It contains most nouns, adjectives, verbs, and derivated adverbs and predicatives. In RUSTWOL, the most productive nouns and adjectives can occur as first parts of compounds. The following examples of possible combinations are presented in surface forms, except for #, which is a sign of word boundary:

REL A + REL A	motorno#-parusnyj	(C1 + Q2+F1)
QUAL A + QUAL A	barhatisto#-mohnatyj	(C2 + Q1+F1)
N + N	stroj#bank	(C3 + Q3)
N + PRES PART ACT	gazo#obrazuu1s2ij	(C4 + Q4)
QUAL A + REL A	geroic1eski#-nezemnoj	(C5 + Q2+F1)
N + N	kvartiro#sdatc1ik	(B1 + Q3)
N + PRES PART ACT	luc1e#ispuskau1s2ij	(B2 + Q4)
N + PAST PART PASS	gazo#zas2is2ennyj	(B3 + Q5)

Some of the first parts in Stem1 cannot be used as independent words, for example the following:

ae1ro#s1kola	(C3 + Q3)
gamma#-kvantovyj	(C4 + Q2+F1)

Lexicon Stem2 includes color adjectives. On the one hand, when two color adjectives are combined, there must be a linking element and a hyphen between the parts. On the other hand, only a linking element is needed, when color adjectives are connected to nouns or adjectives in Stem1.

Lexicon Stem3 contains, firstly, pronouns, numerals, proper nouns, abbreviations and non-inflecting parts of speech that are not formed in the declension types of adjectives. In addition, some nouns, adjectives and verbs that are not partaking in productive compound formation are included in this lexicon. Only some pronouns and numerals can occur as first parts of compounds. Lexicon Stem4 contains only numbers 0...9. The continuation classes of numbers account for words like 0, 125, 2.2, 334, 5, 1997-2000, 50-letie. They also include a continuation to Stem1.

15.4 Final Remarks

The most difficult problem that I faced in developing the first version of the RUSTWOL lexicon and rules was the problem of handling compounds, both parts of which are inflected. This problem is not discussed by me in the document Vilkki 1997. Russian has a fairly productive means of forming compounds by inflecting the both parts in the same case and number. Most of these are nouns, but it is also possible to form compound relative adjectives using this kind of compounding. For example, the dictionary of Russian compounds Bukčina and Kalakučkaja 1987 lists 82,000 compounds, and approximately 5,800 these represent compounds, both parts of which are inflected. Here are some examples of these kinds of compounds in the genitive case:

pisatela1-gumanista 'writer-humanist'
funkcii-kriterija 'function-criterion'
z1ens2iny-uc1enogo (sekretara1) 'woman-scientific (secretary)'

Besides genitive, this kind of inflection concerns all the other singular and plural cases. Because it was difficult to find any appropriate way to handle these kinds of compounds adequately, they were totally excluded from the lexicon.

At a more general level, Koskenniemi (1983) understood that his initial two-level model had significant limitations in handling various kinds of non-concatenative morphotactic processes. Several kinds of non-concatenative phenomena are considered in, for example, Beesley and Karttunen (2003:375-420). They rightly state that non-concatenative morphotactis is the cutting edge of computational morphology. I would like to emphasize, however, that the version of RUSTWOL presented here is not the current one. As far as I know, the current RUSTWOL at Lingsoft has, on the whole, a more adequate system of forming compounds. This newer version represents a new kind of format the practical implementation of which is based on suggestions of Koskenniemi.

Appendix

This appendix gives Cyrillic translations of the alphabet used in RUSTWOL. It also lists all the tags contained in RUSTWOL.

Alphabet:

a	a1	b	c	c1	d	e	e1	f	g	h	i	j	k	l
а	я	б	ц	ч	д	е	э	ф	г	х	и	й	к	л
m	n	o	p	r	s	s1	s2	t	u	u1	v	y	z	z1
м	н	о	п	р	с	ш	щ	т	у	ю	в	ы	з	ж
‘	$													
ь	ъ													

RUSTWOL tag set:

ACC	accusative
ACT	active
ADV	adverb
ADV-CMP	comparative form of adverb
CARD	cardinal number
CMP	comparative
COLL	collective
COMP	compound
CONJ	conjunction
CONST	constituent
DAT	dative
DEF	definite
DEM	demonstrative
FE	feminine
FUT	future
GEN	genitive
IMPERS	impersonal
IMPF	imperfective
IMPV	imperative
INF	infinitive
INDECL	indeclinable
INDEF	indefinite
INSTR	instrumental
INTERJ	interjection
INTERR	interrogative
MA	masculine
N	noun
NOM	nominative
NE	neuter
NEG	negative
NUM	numeral
ORD	ordinal number
PARENTH	parenthetical
PART	participle
PASS	passive
PAST	past tense (preterite)
PCLE	particle
PERF	perfective
PERS	personal
PL	plural
PL1	1st person, plural
PL2	2nd person, plural
PL3	3rd person, plural
POSS	possessive
PRED	predicative
PREP	prepositional
PRES	present tense
PRON	pronoun
REFL	reflexive
SG	singular
SG1	1st person, singular
SG2	2nd person, singular
SG3	3rd person, singular
SH FE	short feminine
SH MA	short masculine
SH NE	short neuter
SH PL	short plural
SUP	superlative
V	verb
V ADV	verbal adverb

Acknowledgements

I wish to thank Kimmo Koskenniemi, Fred Karlsson, Arto Mustajoki and Jouko Lindstedt for the helpful advice on various aspects of the first version of RUSTWOL.

References

Beesley, K., Karttunen, L. 2003. *Finite State Morphology*. Stanford: CSLI Publications.

Bukčina, B.Z., Kalakučkaja, L.P. 1987. *Slitno ili razdel'no*. Moscow: Russkij jazyk.

Jevgen'eva, A.P. (ed.) 1981-1984. *Slovar' russkogo jazyka i-iv*. 2nd ed. Moscow: Russkij jazyk

Kahla, M. (ed.) 1984. *Neuvostoliittolaisten henkilönnimien opas*. Helsinki: Valtion painatuskeskus.

Kahla, M. (ed.) 1982. *Neuvostoliiton paikannimet*. Helsinki: Valtion painatuskeskus.

Kopotev, M, Mustajoki, A. 2003. Principy sozdanija Hel'sinkskogo annotirovannogo korpusa russkih tekstov (HANKO) v seti internet. *Naučno-tehničeskaja informacija*, ser. 2, No. 6: 33-37-

Koskenniemi, K. 1983. *Two-level Morphology: A General Computational Model for Word-form Recognition and Production*. University of Helsinki: Publications of the Department of General Linguistics. No. 11.

Koskenniemi, K.1997. Representations and Finite-State Components in Natural Language. *Finite-State Language Processing*, ed. E. Roche, Y. Schabes, 99-116. Cambridge: The MIT Press.

Kotelova, I.Z. (ed.) 1984. *Novye slova i značenija: slovar'-spravočnik po materialam pressy i literatury 70-h godov*. Moscow: Russkij jazyk.

Kuznecova, A.I., Efremova, T.F. (ed.) 1986. *Slovar' morfem russkogo jazyka*. Moscow: Russkij jazyk.

Paile, Alexander 2003. *Avtomaticeskij analiz russkogo teksta*. Master thesis, University of Helsinki.

Scheitz, E. 1986. *Dictionary of Russian Abbreviations*. Berlin: Veb Verlag Technik.

Švedova, N.J. (ed.) 1982. *Russkaja grammatika i-ii*. Moscow: Nauka.

Tihonov. A.N. 1985. *Slovoobrazovatel'nyj slovar' russkogo jazyka*. Moscow: Russkij jazyk.

Vilkki, Liisa 1997. *RUSTWOL: a System for Automatic Recognition of Russian words*. A technical document, Lingsoft.

Zalizniak, A.A. 1987. *Grammatičeskij slovar' russkogo jazyka*. 3rd ed. Moscow: Russkij jazyk.

Zasorina, L.N. (ed.) 1977. *Častotnyj slovar' russkogo jazyka*. Moscow: Russkij jazyk.

16

Combining Regular Expressions with Near-Optimal Automata in the FIRE Station Environment

BRUCE W. WATSON, MICHIEL FRISHERT AND LOEK CLEOPHAS

We discuss a method for efficiently computing deterministic Brzozowski (derivatives) automata. Our approach is based on efficiently storing regular expressions using parse trees and expressions using common subexpression elimination.

16.1 Introduction

Derivatives of regular expressions were first introduced by Brzozowski in (Brzozowski, 1964). By recursively computing all derivatives of a regular expression, a deterministic automaton can be constructed. To guarantee convergence of this process, derivatives are compared modulo *similarity*, i.e. modulo associativity, commutativity, and idempotence of the union operator. Additionaly, through simplification based on the identities for regular expressions, the number of derivatives can be further reduced.

We have developed an efficient method for computing such automata by combining parse trees with the automata. In our implementation, we recognize and remove similar regular expressions through *global common subexpression elimination* (GCSE) on the parse tree. The concept of GCSE is a well-known optimization technique in the field of compilers, see for example (Cocke, 1970). Because the regular expressions are stored in parse trees,

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

and subtrees of common expressions are reused optimally, we can avoid the expense of storing an entire regular expression (as a string or parse tree) per derivative. Also, we never compute derivatives twice for a class of similar regular expressions.

Reduction of the number of regular expressions by identities is done through regular expression rewriting. Due to the generic framework for rewriting, we are able to reduce using additional rewrite rules, which results in smaller automata.

An earlier version of some of the research reported on in this paper was presented as a poster paper at CIAA 2004 (Frishert and Watson, 2004).

16.1.1 Historical note by Bruce Watson

These algorithms, data-structures, and techniques have now been implemented in the FIRE Station environment, also described in a number of recent articles. The FIRE Station, related to the FIRE Engine series of toolkits for finite automata and regular expressions, is a workstation-type environment (in software) manipulating regular expressions, finite automata, and other finite-state objects, including their languages. In the mid-1990's, I made several visits to Kimmo in Helsinki. The need and underlying ideas for FIRE Station grew directly out of those brainstorming sessions with Kimmo and his group. There were already a number of tools (notably tools from Xerox and AT&T and also INTEX) available, though all rather tightly coupled to their applications in NLP. The core philosophy behind the FIRE Station is to provide a number of efficient application-neutral algorithms and data-structures. Layered on top of this core will be the option of several 'skins,' providing the look-and-feel of the various application domains for finite state techniques, such as: NLP, modeling of concurrent systems, compiler design, text indexing and hardware design. Each such skin may additionally provide domain-specific operators, views of the automata, etc. Kimmo's ongoing interest and inputs have given a unique NLP perspective on the potential applications of a tool such as FIRE Station. (FIRE Station is being made available—also in source form—for noncommercial use.)

16.2 Preliminaries

Definition 1 [Regular Languages and Regular Expressions] We define regular expressions RE over alphabet Σ and the languages they denote, $\mathcal{L}_{RE} \in RE \rightarrow \mathcal{P}(\Sigma^*)$ as follows:

- $\emptyset \in RE$ and $\mathcal{L}_{RE}(\emptyset) = \emptyset$
- $\varepsilon \in RE$ and $\mathcal{L}_{RE}(\varepsilon) = \{\varepsilon\}$
- For all $a \in \Sigma, a \in RE$ and $\mathcal{L}_{RE}(a) = \{a\}$

For $E, F \in RE$

- $E|F \in RE$ and $\mathcal{L}_{RE}(E|F) = \mathcal{L}_{RE}(E) \cup \mathcal{L}_{RE}(F)$
- $E \cdot F \in RE$ and $\mathcal{L}_{RE}(E \cdot F) = \mathcal{L}_{RE}(E) \cdot \mathcal{L}_{RE}(F)$
- $E^* \in RE$ and $\mathcal{L}_{RE}(E^*) = \mathcal{L}_{RE}(E)^*$ □

16.3 Parse Trees

The parse tree is a tree based representation of regular expressions. Each node in the tree defines a regular expression based on its children and the operator associated with the node. In contrast with the binary parse trees that are often found in the literature, our parse trees are n-ary trees. Nodes in the parse tree are represented by the set V, and each node $v \in V$ is either an internal node (has children), or a leaf node.

Definition 2 [The Set of Regular Operators] We define the set of constants and operations on regular languages by their names:
$operators = \{[\emptyset], [\varepsilon], [\Sigma], [*], [|], [\cdot]\}$ □

Definition 3 [Regular Operator Nodes] The set of nodes V is partitioned over $operators$:

- $(\forall i : i \in operators : V_i \subseteq V)$
- $(\cup i : i \in operators : V_i) = V$
- $(\cap i : i \in operators : V_i) = \emptyset$ □

Definition 4 [Structure of the Parse Tree] The structure of the parse tree is uniquely determined by the following four functions:

- $symbol : V_{[\Sigma]} \rightarrow \Sigma$.
- $term : V_{[*]} \rightarrow V$
- $termset : V_{[|]} \rightarrow \mathcal{P}(V)$
- $termlist : V_{[\cdot]} \rightarrow V^*$ □

Definition 5 [Parse Tree to Regular Expression] For a node $v \in V$, we define a mapping $regex \in V \rightarrow RE$, from parse tree to regular expression straightforwardly as:

- $v \in V_{[\emptyset]} \Rightarrow regex(v) = \emptyset$
- $v \in V_{[\varepsilon]} \Rightarrow regex(v) = \varepsilon$
- $v \in V_{[\Sigma]} \Rightarrow regex(v) = symbol(v)$
- $v \in V_{[*]} \Rightarrow regex(v) = term(v)^*$
- $v \in V_{[|]} \Rightarrow regex(v) = (|w : w \in termset(v) : w)$
- $v \in V_{[\cdot]} \Rightarrow regex(v) = termlist(v)_0 \cdot \ldots \cdot termlist(v)_{|termlist(v)|-1}$ □

Definition 6 [Regular Language of a Node $\mathcal{L}_{PT}(v)$] For a node $v \in V$, the regular language represented by v is given by $\mathcal{L}_{PT}(v) \in V \rightarrow RE$ as follows:
$\mathcal{L}_{PT}(v) = \mathcal{L}_{RE}(regex(v))$ □

Definition 7 [Creating Regular Expression Nodes] We can create new nodes in the parse tree through the function $mknode \in ([\emptyset] \cup [\varepsilon] \cup ([\Sigma] \times \Sigma) \cup ([*] \times V) \cup ([|] \times \mathcal{P}(V)) \cup ([\cdot] \times V^*)) \rightarrow V$. This function has to satisfy the following specification:

- $\mathcal{L}_{PT}(mknode([\emptyset])) = \emptyset$
- $\mathcal{L}_{PT}(mknode([\varepsilon]))) = \{\varepsilon\}$
- $\mathcal{L}_{PT}(mknode([\Sigma], a)) = \{a\}, (\forall a \in \Sigma)$
- $\mathcal{L}_{PT}(mknode([*], v)) = \mathcal{L}_{PT}(v)^*, (\forall v \in V)$
- $\mathcal{L}_{PT}(mknode([|], W)) = (\bigcup w : w \in W : \mathcal{L}_{PT}(w)), (\forall W \in \mathcal{P}(V))$
- $\mathcal{L}_{PT}(mknode([\cdot], W)) = \mathcal{L}_{PT}(W_0) \cdot \ldots \cdot \mathcal{L}_{PT}(W_{|W|-1}), (\forall W \in V^*)$ □

Note that these specifications seem weaker than they need to be, however, they allow room for refinement in the implementation. For example, given nodes $v, w \in V$, so that $\mathcal{L}_{PT}(v) = \{\varepsilon\}$ and $\mathcal{L}_{PT}(w) = \{a\}$, the function $mknode([|], v, w)$ may return a new node $u \in V_{[|]} \wedge termset(u) = \{v, w\}$; but it may also simply return w. This leaves room for improvements that will be discussed at a later point.

16.4 Derivatives

First, we adapt Brzozowski's definition of derivatives to our parse tree.

Definition 8 Function $\delta \in V \rightarrow RE$ determines whether or not the regular language represented by a node $v \in V$ contains the empty string ε and is defined as:

$\delta(v) = \varepsilon$, if $\varepsilon \in \mathcal{L}_{PT}(v)$
$\delta(v) = \emptyset$, if $\varepsilon \notin \mathcal{L}_{PT}(v)$ □

Definition 9 [Brzozowski Derivatives] For node $v \in V$ and symbol $a \in \Sigma$ the derivatives function $D \in V \times \Sigma \rightarrow RE$ is defined as:

- if $v \in V_{[\emptyset]}$, then $D(v, a) = \emptyset$
- if $v \in V_{[\varepsilon]}$, then $D(v, a) = \emptyset$
- if $v \in V_{[\Sigma]} \wedge symbol(v) = a$, then $D(v, a) = \varepsilon$
- if $v \in V_{[\Sigma]} \wedge symbol(v) \neq a$, then $D(v, a) = \emptyset$
- if $v \in V_{[*]}$, then $D(v, a) = D(term(v), a) \cdot v$
- if $v \in V_{[|]}$, then $D(v, a) = (|u : u \in childset(v) : D(u, a))$
- if $v \in V_{[\cdot]}$, then $D(v, a) =$
  $(D(termlist(v)_0, a) \cdot termlist(v)_1 \cdot \ldots \cdot termlist(v)_{|termlist(v)|-1})$
  $|(\delta(termlist(v)_0) \cdot D(termlist(v)_1 \ldots termlist(v)_{|termlist(v)|-1}, a))$ □

Our goal is to find or create a node in the parse tree that represents the derivative of a given node-symbol pair. To this end, we introduce the function $\Delta \in V \times \Sigma \rightarrow V$. A straightforward Δ could satisfy $regex(\Delta(v, a)) =$

$D(v, a)$. We deviate slightly and only require the weaker condition of $\mathcal{L}_{PT}(\Delta(v, a)) = \mathcal{L}_{RE}(D(v, a))$, i.e. that the regular languages rather than the regular expressions are equivalent. This allows us some room to add optimizations, potentially leading to smaller automata. It also allows for a straightforward definition of Δ in terms of $mknode$, as seen in Def. 10.

Definition 10 [Derivatives via the Parse Tree] We create the function $\Delta \in V \times \Sigma \rightarrow V$, which computes the node representing the derivative of a given node $v \in V$ and symbol $a \in \Sigma$, such that $\mathcal{L}_{PT}(\Delta(v, a)) = \mathcal{L}_{RE}(D(v, a))$. Δ is expressed in terms of $mknode$:

- if $v \in V_{[\emptyset]}$, then $\Delta(v, a) \equiv mknode([\emptyset])$
- if $v \in V_{[\varepsilon]}$, then $\Delta(v, a) \equiv mknode([\emptyset])$
- if $v \in V_{[\Sigma]} \wedge a = symbol(v)$, then $\Delta(v, a) \equiv mknode([\varepsilon])$
- if $v \in V_{[\Sigma]} \wedge a \neq symbol(v)$, then $\Delta(v, a) \equiv mknode([\emptyset])$
- if $v \in V_{[*]}$, then $\Delta(v, a) \equiv mknode([\cdot], \Delta(term(v), a), v)$
- if $v \in V_{[|]}$, then $\Delta(v, a) \equiv mknode([|], \cup u \in V : u \in termset(v) : \Delta(u, a))$
- if $v \in V_{[\cdot]} \wedge termlist(v)_0 \notin null$, then $\Delta(v, a) \equiv mknode([\cdot],$
  $\Delta(termlist(v)_0, a), termlist(v)_1, \ldots, termlist(v)_{|termlist(v)|-1})$
- if $v \in V_{[\cdot]} \wedge termlist(v)_0 \in null$, then $\Delta(v, a) \equiv mknode([|],$
  $\{mknode([\cdot],$
  $\Delta(termlist(v)_0, a), termlist(v)_1, \ldots, termlist(v)_{|termlist(v)|-1}),$
  $\Delta(mknode([\cdot], termlist(v)_1, \ldots, termlist(v)_{|termlist(v)|-1}), a)\})$

$\square$

All that remains is an implementation of the function $mknode$. To this end, we now discuss our means of dealing with similar expressions and reduction via identities.

16.5 Common Subexpression Elimination

The subexpression of a node v is the regular expression as described by the parse tree. It is not uncommon for two equivalent subexpressions to occur in different parts of the parse tree. By finding and eliminating these *common subexpressions*, we can merge similar derivatives.

Definition 11 [Subexpression Equivalence $\sim_{cse}$] Nodes $v, w \in V$ are in relation $v \sim_{cse} w$ holds if any of the following holds:

- $v = w$
- $v, w \in V_{[\emptyset]}$
- $v, w \in V_{[\varepsilon]}$
- $v, w \in V_{[\Sigma]} \wedge symbol(v) = symbol(w)$
- $v, w \in V_{[*]} \wedge term(v) \sim_{cse} term(w)$

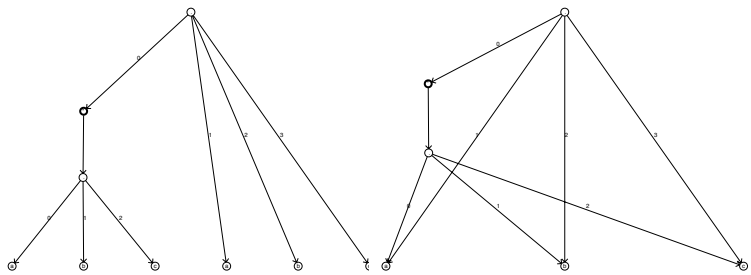


FIGURE 1 (a) $(abc)^*abc$ before GCSE (b) after GCSE

- $v, w \in V_{[|]} \wedge (\forall p \in termset(v) : (\exists q \in termset(w) : p \sim_{cse} q)) \wedge$ $(\forall q \in termset(w) : (\exists p \in termset(v) : p \sim_{cse} q))$
- $v, w \in V_{[\cdot]} \wedge (\forall i : 0 \leq i \leq |termlist(v)| : termlist(v)_i \sim_{cse} termlist(w)_i)$

Note that $v \sim_{cse} w \equiv regex(v) = regex(w)$ □

We can reduce all nodes that are in the same equivalence class defined by $\sim_{cse}$ to a single node. This process is called *Global Common Subexpression Elimination* (GCSE). Removing equivalent nodes does not affect the regular languages represented, however, it does change the parse tree into a directed acyclic graph (DAG). As an example of this, the regular expression *(abc)*abc* results in the parse tree in Figure 1(a). The subexpression *abc* occurs in two locations. We can replace these by a single instance, as in Figure 1(b). Note that we will continue to use the term parse tree, since that is still the intended interpretation of the graph; the fact that it is a DAG merely provides us with a more efficient representation.

If we integrate GCSE into the function $mknode$, we can establish the following invariant:

Definition 12 [CSE Invariant] $(\forall v, w \in V : v \sim_{cse} w \Rightarrow v = w)$ □

This CSE invariant means that we will never create a new node if a $\sim_{cse}$ equivalent node already exists, and it allows us to detect common subexpressions without resorting to expensive recursion:

Definition 13 [Subexpression Equivalence without recursion] Nodes $v, w \in V$, are in relation $v \sim_{cse} w$ if CSE Invariant of Def. 12 holds, and if any of the following holds:

- $v = w$
- $v, w \in V_{[\emptyset]}$
- $v, w \in V_{[\varepsilon]}$
- $v, w \in V_{[\Sigma]} \wedge symbol(v) = symbol(w)$

TABLE 1 Rewrite Rules for identities. Note that $E \in RE$

$$\emptyset \cdot E \rightarrow \emptyset$$
$$E \cdot \emptyset \rightarrow \emptyset$$
$$\varepsilon \cdot E \rightarrow E$$
$$E|\emptyset \rightarrow E$$

- $v, w \in V_{[*]} \wedge term(v) = term(w)$
- $v, w \in V_{[|]} \wedge termset(v) = termset(w)$
- $v, w \in V_{[\cdot]} \wedge termlist(v) = termlist(w)$ □

When attempting to create a new (internal) node with operator $o \in operators$ and children W, finding a node that is $\sim_{cse}$ equivalent (if it exists) can be done by examining the parents for the child nodes in W to find a node $m \in V_o$ and children equal to W. We can instantly find the parents of a particular node by storing the reverse relations from the parse tree. Note that it is sufficient to search the parents of only one of the elements of W for an equivalent parent node, rather than all the children, because a matching node will be parent to all the nodes in W. To maintain the CSE Invariant of Def. 12, we return the equivalent node if it is found to exist, and only create a new node if it does not exist.

16.6 Rewriting

As suggested by Brzozowski (Brzozowski, 1964), the number of derivatives can be reduced by simplification using the identities. We implement this using a rewriting system as described in (Frishert et al., 2003). As discussed, the specification of Δ was deliberately weak, which now allows us to use any number of rewrite rules. If we wish to obtain the exact Brzozowski derivatives automata, we restrict ourselves to the rewrite rules in Table 1. If we add additional rewrite rules we can potentially obtain smaller automata.

Combining rewriting and GCSE, the function $mknode$ can now be implemented as follows for a given operator and operand (either a symbol, node, nodeset or a nodelist): If an applicable rewrite rule exists, apply that rule, resulting in a new operator/operand pair. Repeat this until there are no further applicable rewrite rules. For the final operator/operand pair, we search the existing nodes for a CSE-equivalent node. If such a node exists, we return that node; otherwise we add a new node to V and set its operator/operands accordingly.

16.7 Results and Future Work

We have implemented the approach discussed in this paper in our tool FIRE STATION, see (Frishert, 2005). All figures in this paper were generated using

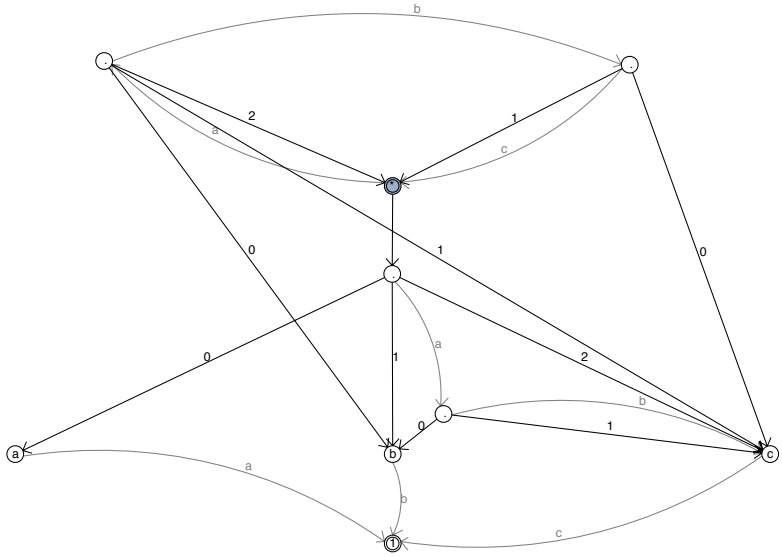


FIGURE 2 Combined Parse Trees for the Derivatives of $(abc)^*$

FIRE STATION. In Figure 2, the combined parse graph for the derivatives of $(abc)^*$ is shown. The numbered edges indicate the order of concatenated nodes: due to GCSE, a node can be used in multiple concatentations, and the order for these concatentations is sometimes conflicting, making it impossible for the concatenated nodes to be drawn in left-to-right order.

The extended regular operators: negation, intersection, relative/symmetric difference, negation, as well as the POSIX character classes, and repeat ranges can easily be added to this framework and require no special treatement.

The approach we have discussed in this paper also lends itself well to partial derivatives (Antimirov, 1996), which also have been implemented successfully in FIRE STATION.

We see two interesting next steps. First, additional rewrite rules, which may result in further reduction of automata sizes, could be included. Second, it may be possible to perform incremental minimization, reducing intermediate memory requirements.

References

Antimirov, V. 1996. Partial derivatives of regular expressions and finite automata

constructions. *Theoretical Computer Science* 155:291–319.

Brzozowski, J.A. 1964. Derivatives of Regular Expressions. *Journal of the ACM* 11(4):481–494.

Cocke, J. 1970. Global common subexpression elimination. In *Proceedings of a symposium on compiler optimization*, pages 20–24.

Frishert, Michiel. 2005. *FIRE Station: a FInite automata & Regular Expression playground*. Master's thesis, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven.

Frishert, M., L. Cleophas, and B.W. Watson. 2003. The effect of rewriting regular expressions on their accepting automata. In *Proceedings of the 8th International Conference on Implementation and Application of Automata (CIAA 2003)*, vol. 2759 of *Lecture Notes in Computer Science*, pages 304–305. Springer.

Frishert, M. and B.W. Watson. 2004. Combining regular expressions with (near-) optimal Brzozowski automata. In *Proceedings of the 9th International Conference on Implementation and Application of Automata (CIAA 2004)*, vol. 3317 of *Lecture Notes in Computer Science*, pages 319–320. Springer.

17

Linguistic Grammars with Very Low Complexity

ANSSI YLI-JYRÄ

17.1 Introduction

Fifteen years ago, in the COLING-90 in Helsinki, Kimmo Koskenniemi sketched a finite-state approach to surface syntax (Koskenniemi, 1990): an approach that later became known by the name Finite-State Intersection Grammar (FSIG). During the subsequent few years this approach was investigated by Koskenniemi's associates Pasi Tapanainen, Atro Voutilainen and some others in the Research Unit of Multilingual Language Technology at the Department of General Linguistics at the University of Helsinki.

A while after Koskenniemi's proposal, technical problems related to the state complexity of FSIG grammars became a major challenge in the further development of the system. However, this was largely due to the fact that the rules in the first grammars did not suggest means to exploit the locality of linguistic constraints. Meanwhile, a similar but less ambitious constraint system flourished independently in France as Maurice Gross and his students had introduced *local grammars* and developed algorithms that apply these to lexically ambiguous sentences.

In 1995, the current author became involved, for the first time, in investigations that pursued more efficient FSIG parsing strategies and complexity. These investigations continued in 2000's and led to a PhD thesis. This chapter tries to give an overview of the recent discoveries related to the complexity of FSIG parsing. The chapter is structured as follows: Section 17.2 sketches a rough background of Kimmo's approach, relating FSIG to the well known CG framework, and to finite-state methods in general. Section 17.3 general-

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

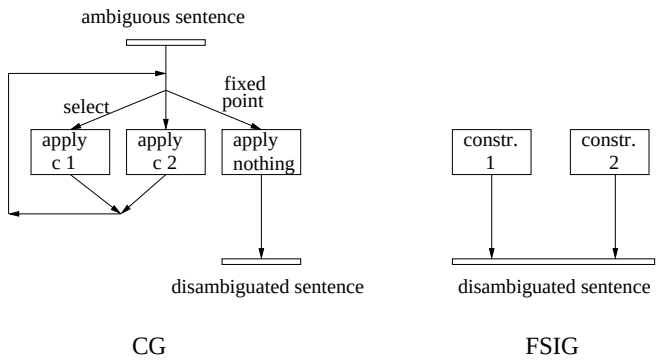


FIGURE 1 FSIG and its forerunner: Constraint Grammar.

izes the FSIG architecture to non-regular languages and linguistically important structures. Section 17.4 explains the star-freeness property of the FSIG grammars. Section 17.5 approaches FSIG parsing through a layered structure, essentially improving the compactness of the grammar.

17.2 The Background of FSIG

17.2.1 The Inspiration

Kimmo's FSIG approach was largely inspired by the Constraint Grammar (CG) system (Karlsson, 1990) that can be described as follows.

The input of CG is a tokenized sentence with alternative readings listed at each token. Each CG constraint rule application is a transformation that removes one or more readings of an ambiguous token in a given context. Each transformation is a rational transduction. The context conditions tested by the rules are able to refer to contextual ambiguity and to test bunches of alternative readings in the token and its context.

As a whole, a CG parser is a combination of a prioritized union of rules that is applied iteratively up to a fixed point where no rule can reduce any more ambiguity. The parser works by iteratively selecting a constraint and an ambiguous token and applying the selected constraint to the selected token. This process terminates: the maximal number of iterations carried out by the parser is proportional to the length of the sentence. Furthermore, the order in which the tokens are processed may require a linear number of back and forth jumps between token positions. It is thus not generally possible to characterize a CG parser (Figure 1, left side) as a regular relation.

In 1983, Kimmo Koskenniemi had became an inventor of a parallel constraint system, the two-level model of morphology (Koskenniemi, 1983) (TWOL). TWOL had already been proved a practical alternative to a cascade

of phonological rules, such as used in generative phonology. As a natural continuation to this work, Kimmo proposed a similar approach to syntactic parsing and disambiguation: a system of parallel finite-state constraints that defined a regular relation (Koskenniemi, 1990). This system presented an alternative for the serial approach of the CG parsing, but was not claimed to be equivalent to it.

17.2.2 FSIG as a One-Level System

The formal elegance of FSIG was remarkable. It was a one-level system, while, in contrast, the number of intermediate levels in CG was not bounded by a constant. Furthermore, FSIG was able to give new insight on possible parsing approaches by putting into practice a set-theoretic semantics for grammars. FSIG parsing consists of two phases:

1. generation of a set of potential reading strings, and
2. constraint-driven selection of the grammatical readings.

First, potential *sentence readings* of the input sentence — each being a string of morphemes and word boundaries — are constructed by inserting annotation codes freely into the sequence of input tokens (in practice, it is desirable that the insertion is controlled by lexical lookup). This creates a set of alternatives that is often referred to as an *ambiguous sentence*[1]. The generated set is represented by a deterministic finite automaton (DFA) that is often called the *sentence automaton*[2].

Second, there are *constraint automata* each of which implements a linguistic or administrative constraint, originally described using an extended FSIG notation of regular expressions. When a new ambiguous sentence has been generated the constraint automata are applied to reduce ungrammatical sentence readings (strings) in the sentence automaton. This can be carried out, in theory, by computing a direct product of the sentence automaton and the constraint automata, or by performing a backtracking search for alternative analyses. The direct product automaton describes exactly those sentence readings that are recognized by every constraint automaton.

In contrast to the fixed point semantics of the CG disambiguation process, the standard FSIG is a one-level constraint system with hard constraints: if some constraint rejects all potential readings, there will be no readings left in the output. An alternative FSIG framework with soft constraints would be desirable for purposes of robust parsing (although we do not want to run into the practical difficulties of Optimality Theoretic approaches).

[1] Generation of this set resembles the GEN function in Optimality Theory, but can be already constrained by the lexicon.

[2] Earlier, this automaton used to be acyclic, but this restriction is no more maintained in recent FSIG systems.

17.2.3 FSIG and State Complexity

In addition to the parallel constraints, the initial developers of the FSIG framework adopted a useful operation from the formalism of the original two-level morphology: the so-called context restriction operator became a part of the FSIG notation. This regular operator has an interesting history, and it allows for further generalizations (Yli-Jyrä and Koskenniemi, 2004).

The FSIG rule formalism is able to specify complex finite-state grammars in a very compact fashion. In fact, we could estimate that the deterministic state complexity (the size of the minimal DFA) corresponding to the combination of all constraint automata of a full-coverage grammar could be some 10^{10} – 10^{1000} states. The constraints can be applied in linear time to the input sentence, according to the the input size, but linear time complexity alone does not thus imply a practical implementation.

It is also important to understand how the complexity of the grammar is related to the way the grammar is designed. A few initial results on state complexity of *bracketing restriction operator* – a recent novelty (Yli-Jyrä, 2003c) – and *context restriction operator* (Yli-Jyrä and Koskenniemi, 2004) have been published. Their complexity grows, in the worst case, exponentially according to the maximum depth of bracketing.

17.2.4 The Quest for Locality in FSIG

Some FSIG experts, including Kimmo himself, have always maintained the optimism that an efficient parser for FSIG could be found. The hope is motivated by the fact that the parse result – the reduced set of alternatives – does not exhibit remarkable state complexity although its computation is difficult. To be more successful, an efficient parser would need to decompose the grammar in a fashion that maintains compactness during the intermediate parsing steps. How this should be done has been an open problem, but a recently presented compilation method for rules (Yli-Jyrä and Koskenniemi, 2004) sheds some light on how the grammar can be split into almost independent modules.

An informal comparison to a personal computer may be helpful in understanding compact representations of finite automata and transition functions: CPUs implement predefined state transitions in an immense state space, without any difficulties. This is possible because (i) the CPUs modify, within one step, only a small portion of the computer's memory, making only *local state transitions* at a time, and (ii) the next state is often computed in parallel, largely independent circuits within the processor (such as the program counter, the arithmetic logic unit and the cache). If similar design principles – locality and decomposition – could be used to store the FSIG grammar and the intermediate results, we could perhaps find an efficient parser.

17.3 A generalization of Kimmo's Approach

17.3.1 Anti-Approximations

Constraint grammars (CGs) and local grammars are typically applied to a flat sequence of lemmas and tags, without any attempts to cope with bracketed trees. In contrast to this, Kimmo's proposal and the first FSIGs included clausal embedding up to one level of clause boundaries. It was argued by Kimmo that only a tiny proportion of running-text sentences would contain a double-center-embedded clause.

The current author (Yli-Jyrä, 2003a) considered explicitly an arbitrary limit d for center embedding in FSIG. This generalization suggests the possibility of taking union of the languages of an FSIG grammars G when its d-parameter goes to infinite:

$$L(\hat{G}) = \cup_{d=0..\infty} L(G_d) = \lim_{d\to\infty} L(G_d).$$

According to the formula, every FSIG grammar G, such as Voutilainen's English grammar (Voutilainen, 1997) is in fact a parameterized specification that gives us both

- a series of finite-state grammars $G_0, G_1, G_2, \ldots$, and
- an idealistic generalization, an *anti-approximation* $L(\hat{G})$.

In the case of Voutilainen's English grammar, the anti-approximation $L(\hat{G})$ is context-free, but, in some other cases, it can be non-context-free[3].

This view suggests a perspective on how non-regular grammars could be learned: through a series of regular languages. Furthermore, the view suggests connections to bracket-based representations of non-regular grammars.

17.3.2 Chomsky-Schützenberger Representations

In early 1960's, Noam Chomsky and Marcel Paul Schützenberger (1963) discovered a technique to represent the language of any context-free grammar G as a homomorphic image of an intersection of a Dyck language and a regular language that depends on G.

FSIG grammars have a close relationship to the Chomsky-Schützenberger representations. In any FSIG grammar G_d, the parameter d actually specifies an approximation of a Dyck language. By using, in the constraint semantics, a Dyck language instead of its regular approximation, we get a representation for the anti-approximated language $L(\hat{G})$.

This view has been very fruitful. We have been able to specify many new Chomsky-Schützenberger style representations[4] for various classes of

[3] This is possible if the grammar uses crossing sets of brackets as in (Yli-Jyrä, 2004).

[4] In some of our representations, the Dyck language is distributed and used as a constant in the rules. We refer to them loosely as Chomsky-Schützenberger style representations.

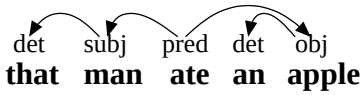


FIGURE 2 A dependency tree.

formal grammars. There are such representations for (i) extended context-free languages, (ii) projective dependency grammars, and (iii) certain mildly context-sensitive grammars (MCSGs) that correspond to some families of non-projective dependency grammars.

A tantalizing opportunity of this approach is to try and develop similar bracketing-based representations to further examples of MCSGs, such as tree-adjoining grammars and (multi-modal) combinatorial categorial grammars. Whether this can be done is an open problem, but a success would greatly increase the relevance of the FSIG framework to Natural Language Processing (NLP), since a single architecture would allow for both an idealistic generalization and a series of finite-state approximations.

17.3.3 A New Bracketed Representation for Dependencies

According to Koskenniemi (1990), his approach *does not aim to uncover semantically oriented distinctions*. This limitation was maintained in the first FSIG systems that were clearly meant for partial parsing and not for, *e.g.* producing an explicit dependency structure.

Dependency links indicate, ideally, how words with predicate-argument structures are composed in a semantically coherent way. As we saw in the above, recent developments of FSIG have introduced new sub-frameworks, including frameworks for projective and non-projective dependency parsing.

For example, the dependency tree in Figure 2 can be represented as a bracketed string as in Figure 3.

#			**that**	det	$[_{\leftarrow}$	
#	$]_{\leftarrow}$		**man**	subj	$[_{\leftarrow}$	
#	$]_{\leftarrow}$		**ate**	pred	$[_{\rightarrow}$	
#			**an**	det	$[_{\leftarrow}$	
#	$]_{\leftarrow}$	$]_{\rightarrow}$	**apple**	obj		#

FIGURE 3 String with dependency-tree bracketing. The line breaks have been added.

In non-projective dependency structures (Yli-Jyrä, 2003b, 2004), we use disjoint sets of brackets and follow a non-trivial generalization of the so-called stack discipline when allocating crossing brackets. Thanks to this, each non-projective structure has a unique encoding as a bracketed string. The corresponding system of stacks is connected to a class of MCSGs (Yli-Jyrä and

Nykänen, 2004).

The ability of recent FSIGs to cope with dependency trees (and graphs) under certain performance restrictions suggests that the framework might be capable of assigning even some semantically coherent structures in terms of syntactic dependencies.

17.4 A Characterization of the Complexity of FSIG

17.4.1 Background

The *star-free languages* are the smallest class of languages that contains all finite languages and is closed under concatenation and the Boolean operations.

In Coding Theory, Schützenberger (1965) made a seminal finding by characterizing the star-free languages with aperiodic finite syntactic monoids[5]. Mathematics of Coding Theory and in particular the study of star-free languages are inherently connected to linguistic performance, communication and error tolerance, but star-free languages are seldom discussed in linguistic literature. As a positive example, Kornai (1985) argues on practical limits in natural language semantics, and how this supports an assumption of star-freeness of natural language. The relevance of star-free languages to the language acquisition task has been demonstrated separately by learning algorithms that cope with certain star-free classes of regular languages (Segarra et al., 2003).

17.4.2 Establishment of Star-Free FSIGs

The property of star-freeness has been recently assigned to the FSIG framework. First, the star-freeness of the annotated language described by Voutilainen's English FSIG was established through a rewriting approach (Yli-Jyrä, 2003a). Second, it has become increasingly clearer that the star-freeness restriction does not imply essential losses in the linguistic applicability of FSIGs although it is not difficult to construct artificial examples of FSIGs that fail to be star-free. This is indicated by the flavors of star-free FSIGs that coped with various syntactic structures, including bracketed string representations for unranked constituent trees, projective dependency trees and restricted non-projective dependency trees.

17.4.3 Definability in the First-Order Logic

Robert McNaughton and Seymour Papert (1971) discovered that star-free languages are exactly those described in $FO[<]$, a fragment of first-order logic whose signature contains linear order relation $<$ over string positions. This result is important because it connects star-free languages, such as described

[5]Transitions of a minimal DFA $A = (Q, i, F, \Sigma, \delta)$ define for $w \in \Sigma^*$ the function $\delta_w : Q \to Q$. The set of all functions δ_w with a composition operator and the identity element δ_ϵ is the transition monoid of A as well as the syntactic monoid of the language $L(A)$.

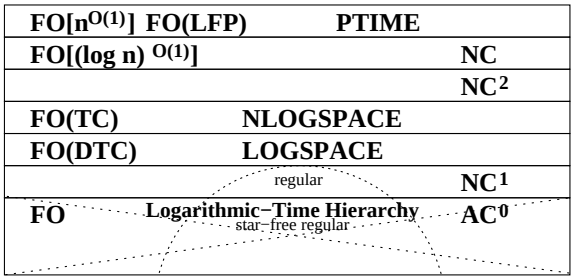


FIGURE 4 Computational complexity of polynomial-time problems, adapted from Immerman (1999).

by FSIG grammars, to (i) the locality characterizations of first-order definable structures, and (ii) to descriptive complexity.

First, we get access to a famous theorem by W. Hanf (Immerman, 1999, p.102-103). According to this theorem, first-order formulae with a *bounded quantifier rank*[6] cannot distinguish between two graphs of bounded degree if the graphs have the same number of local neighborhoods of all possible types where the number of possible types depends exponentially on the quantifier rank. The definition of locality is here more general than in NLP since it involves quantification.

Second, we get access to results in Finite Model Theory, where many computational complexity classes have been characterized using fragments of first-order logic. The close relationship between the computational complexity of problems and the richness of logical language needed to describe them — their descriptive complexity — was established when Ron Fagin showed in 1974 that the problems computable in nondeterministic polynomial time (NP) are exactly characterized by the problems that can be described in existential second-order logic. Neil Immerman (1999, p.2) summarizes the role of descriptive complexity as follows:

> *It [descriptive complexity] gives a mathematical structure with which to view and set to work on what had previously been engineering questions.*

17.4.4 Parallel Computational Complexity

When the languages definable with $FO[<]$ are placed into the picture of computational complexity classes, we observe that they correspond, as illustrated in Figure 4,

- to the logarithmic-time hierarchy, and
- to the uniform circuit complexity class AC^0.

[6]The quantifier rank of a first-order formula is basically the number of nested quantifiers.

A short explanation for some classes in Figure 4 is in place. The *logarithmic-time hierarchy* (LH) contains languages that can be recognized with an alternating Turing machine (ATM) in *logarithmic time* (according to the length of input) using a bounded number of alternations between existential and universal states. The *circuit complexity class* AC^0 consists languages whose strings can be recognized using a constant depth, unbounded-fan-in polynomial-size AND-OR circuits. The circuits in the class NC^1 differ from AC^0 by having a logarithmic depth according to the length of the strings, but restricting the AND and OR gates to ones with two fan-ins.

Star-freeness implies an essential restriction to the parallel computational complexity and circuit complexity of regular languages. Among all regular languages, there are some that do not belong to AC^0, but all are included in NC^1. AC^0 contains all star-free regular languages (Thomas, 1997).

17.5 Structure of Annotated FSIG Languages

17.5.1 The Dot-Depth Hierarchy

Based on the star-freeness of FSIGs, we are able to study the means to represent and parse these grammars in a compact fashion. Star-free languages admit representations that are not available to regular languages in general.

A particularly interesting representation of star-free languages is based on the closure of finite languages, Boolean operations and so-called *concatenation products*. Such a representation of star-free languages generates an infinite sequence or hierarchy of language classes. One of the possible sequences is the dot-depth hierarchy $\mathbf{B}_0, \mathbf{B}_1, \mathbf{B}_2, \ldots$ that was introduced by Brzozowski and Knast (1978). It defines the set of all star-free languages:

$$SF = \cup_{i=0\ldots\infty}\mathbf{B}_i = \lim_{i\to\infty} \mathbf{B}_i.$$

The dot-depth hierarchy is defined over an alphabet Σ as follows:

- $\mathbf{B}_0$ consists of finite and co-finite subsets of Σ^*,
- $\mathbf{C}_i$ consists of concatenations of languages in $\mathbf{B}_i$,
- $\mathbf{B}_i$ consists of Boolean combinations of languages in $\mathbf{C}_{i-1}$.

Thomas (1982) showed that the dot-depth hierarchy corresponds to the quantifier-alternation and logarithmic-time hierarchies mentioned above. According to Thomas, language $L \in \mathbf{B}_i$ can be described by a prenex normal-form that has a so-called Σ_i prefix of quantifiers[7].

If we knew the lowest dot-depth level $\mathbf{B}_i$ that contains the language (the set of annotated strings) of a grammar, we could say more about the parallel computational complexity of the grammar. Unfortunately, the determination

[7] A formula is in prenex normal-form if it consists of a string of quantifiers applied to a quantifier free formula.

of the exact dot-depth of a language is a difficult (open) problem. We can still easily approximate the level from above because the dot-depth depends mainly on d, the depth of allowed bracketing. For example, approximations of the Dyck language can be constructed using a recursive star-free formula (Yli-Jyrä, 2003a) that specifies a language belonging to $\mathbf{B}_{O(d)}$. This can be shown easily by the structure of the recursive formula.

In summary, understanding of the locality in FSIG grammars can be largely built around the relationship between descriptive and parallel computational complexity, d, the dot-depth and the size of the quantifier prefix.

17.5.2 Relative State Complexity of FSIGs

Parallel computational complexity of FSIGs is not just about non-deterministic time. If the minimal DFA equivalent to an ATM could be constructed in a straightforward way (the problem is undecidable for arbitrary ATMs), regular operations applied during the construction would contribute to the *state complexity* of the result. It is imaginable that the alternation between the existential and universal states would amount for additional steps in the state complexity.

In an FSIG approximation of non-projective dependency grammars, the depth d of bracketing corresponds to the number of alternations between concatenations and Boolean operations, while the combination of n disjoint sets of brackets corresponds to an intersection of n star-free languages. Both of these parameters are able to cause an exponential growth in the state complexity of some pathological FSIG grammars.

In the structure of FSIG languages, certain language class distinctions seem to differ radically from the Chomsky hierarchy. For example, if we anti-approximate the mentioned FSIG implementations of non-projective dependency grammars, n induces a hierarchy of MCSGs. Such hierarchies of MCSGs are often seen to exhibit a competence feature, while limited clausal embedding would be an example of a performance feature. In contrast to this complexity landscape, both these parameters (n crossings and d embeddings) appear to be equally important when we determine the state complexity of an FSIG, and the number of crossing sets of brackets (n) does not have any role, when we determine the asymptotic bound for the dot-depth.

17.5.3 Parallel Decompositions

Each new dot-depth level makes references to lower dot-depth levels in a similar fashion as compact parse forest representations pack ambiguity that is beyond the domain of locality. This observation suggests a compact parallel representation for FSIG grammars where $n = 1$ and $d > 0$ (an FSIG with $n > 1$ is obtained by combining simpler FSIGs under intersection). The use of such a representation requires the following steps:

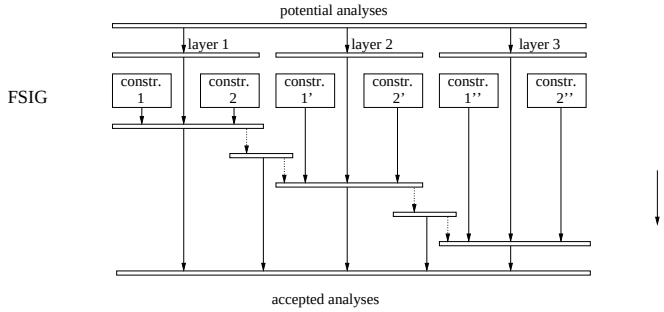


FIGURE 5 FSIG parsing with layers and sub-grammars.

- decomposition of each FSIG constraint into separate constraints each of which checks one layer, *i.e.* level of brackets (>sub-grammars),
- applying the constraints of each layer into a separate copy of the sentence automaton, and
- combining all the constrained sentence automata to obtain the final result.

These tasks have been discussed more in detail by the author (Yli-Jyrä, 2005). A rough overview of the proposed parsing strategy is presented in Figure 5.

17.6 Conclusion

We have presented an overview of the FSIG approach and related FSIG grammars to issues of very low complexity and parsing strategy. We ended up with serious optimism according to which most FSIG grammars could be decomposed in a reasonable way and then processed efficiently.

References

Brzozowski, Janusz A. and Robert Knast. 1978. The dot-depth hierarchy of star-free languages is infinite. *Journal of Computer and System Sciences* 16:37–55.

Chomsky, Noam and Marcel-Paul Schützenberger. 1963. The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg, eds., *Computer Programming and Formal Systems*, pages 118–161. Amsterdam: North Holland.

Immerman, Neil. 1999. *Descriptive Complexity*. Graduate Texts in Computer Science. New York: Springer-Verlag.

Karlsson, Fred. 1990. Constraint grammar as a framework for parsing running text. In H. Karlgren, ed., *13th COLING 1990, Proceedings of the Conference*, vol. 3, pages 168–173. Helsinki, Finland.

Kornai, András. 1985. Natural language and the Chomsky hierarchy. In *2nd EACL 1985, Proceedings of the Conference*, pages 1–7. Geneva, Switzerland.

Koskenniemi, Kimmo. 1983. *Two-level morphology: a general computational model for word-form recognition and production*. No. 11 in Publications of the Department of General Linguistics, University of Helsinki. Helsinki: Yliopistopaino.

Koskenniemi, Kimmo. 1990. Finite-state parsing and disambiguation. In H. Karlgren, ed., *13th COLING 1990, Proceedings of the Conference*, vol. 2, pages 229–232. Helsinki, Finland.

McNaughton, Robert and Seymour Papert. 1971. *Counter-Free Automata*. No. 65 in Research Monograph. Cambridge, Massachusetts: MIT Press.

Schützenberger, Marcel Paul. 1965. On finite monoids having only trivial subgroups. *Information and Computation (Information and Control)* 8(2):190–194.

Segarra, E., E. Sanchis, F. García, L. Hurtado, and I. Galiano. 2003. Achieving full coverage of automatically learnt finite-state models. In *Proceedings of the Workshop on Finite-State Methods in Natural Language Processing*, pages 135–142. Agro Hotel, Budapest.

Thomas, Wolfgang. 1982. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences* 25:360–376.

Thomas, Wolfgang. 1997. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages*, pages 389–455. Springer.

Voutilainen, Atro. 1997. Designing a (finite-state) parsing grammar. In E. Roche and Y. Schabes, eds., *Finite-State Language Processing*, chap. 9, pages 283–310. Cambridge, MA, USA: A Bradford Book, the MIT Press.

Yli-Jyrä, Anssi Mikael. 2003a. Describing syntax with star-free regular expressions. In *11th EACL 2003, Proceedings of the Conference*, pages 379–386. Agro Hotel, Budapest, Hungary.

Yli-Jyrä, Anssi Mikael. 2003b. Multiplanarity – a model for dependency structures in treebanks. In J. Nivre and E. Hinrichs, eds., *TLT 2003. Proceedings of the Second Workshop on Treebanks and Linguistic Theories*, vol. 9 of *Mathematical Modelling in Physics, Engineering and Cognitive Sciences*, pages 189–200. Växjö, Sweden: Växjö University Press.

Yli-Jyrä, Anssi Mikael. 2003c. Regular approximations through labeled bracketing (revised version). Submitted to the post-proceedings of the 8th Formal Grammar conference, Vienna, Austria, 16–17 August, 2003. A link available at `http://www.ling.helsinki.fi/~aylijyra/dissertation/`.

Yli-Jyrä, Anssi Mikael. 2004. Axiomatization of restricted non-projective dependency trees through finite-state constraints that analyse crossing bracketings. In G.-J. M. Kruijff and D. Duchier, eds., *Proc. Workshop of Recent Advances in Dependency Grammar*, pages 33–40. Geneva, Switzerland.

Yli-Jyrä, Anssi Mikael and Kimmo Koskenniemi. 2004. Compiling contextual restrictions on strings into finite-state automata. In L. Cleophas and B. W. Watson, eds., *The Eindhoven FASTAR Days, Proceedings*, no. 04/40 in Computer Science Reports. Eindhoven, The Netherlands: Technische Universiteit Eindhoven.

Yli-Jyrä, Anssi Mikael and Matti Nykänen. 2004. A hierarchy of mildly context sensitive dependency grammars. In G. P. Gerhard Jäger, Paola Monachesi and S. Wintner, eds., *Proceedings of the 9th conference on Formal Grammar 2003 "FGNancy"*.

Yli-Jyrä, Anssi Mikael. 2005. *Contributions to the Theory of Finite-State Based Linguistic Grammars*. No. 38 in Publications of the Department of General Linguistics, University of Helsinki. Helsinki: Helsinki University Printing House.

Part II

Speech and Meaning

18

Speech is Special: What's Special about It?

OLLI AALTONEN AND ESA UUSIPAIKKA

From the ethologist perspective, speech is to the human being as echolocation is to the bat or song is to the bird. Thus, speech, as well as the phonological communication it underlies, is plainly a species-typical product of evolution. Speech defined as the production and perception of vowels and consonants originates from a pre-phonetic capacity to perform speech sounds and gestures. Similarly, language defined as the syntax "machine" originates from a pre-syntactic capacity to organize longer sequences of sounds and gestures. We suggest therefore that the faculty of human language is biological and thus a product of evolution. Furthermore we suggest that formal language follows from speech which is based on motion (gestures) and perception of motion (sensorymotor perception of articulatory gestures).

18.1 Biological View of Language and Speech

Language depends on 'being human'. From a scientific perspective language is neither a divine gift nor a "cultural invention", but a product of human biological evolution. Spoken language evolved to make rapid vocal communication possible, providing man with a better chance of surviving in the struggle for existence (Darwin, 1874). Every normal individual acquires language in a uniform and automatic way by going through the same stages at the same age, without requiring specific instruction (Stromswold, 1996). Once learnt, the complex processes of speech production, perception and syntactic coding become automatized and are carried out below the level of conscious awareness, allowing the semantic content of the message to be the primary concern

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

of the speaker or the listener. Hence, in order to say a word the speakers need not to know what sequence of sounds it comprises but only to think of its meaning. Indeed, they do not even have to know that it has a spelling. The specialized speech system automatically converts the phonological representation of the word into the coarticulated movements of the articulators that convey it. Correspondingly, to perceive a word, listeners need not puzzle out the complex and peculiarly phonetic relation between signal and the phonological message it conveys. All we have to do to perceive speech is to listen; somehow the meanings just emerge as the sounds go by. Again, the phonetic specialization automatically parses the sound so as to recover its phonetic structure. Hence processes of speech, whether in production or perception, are not calculated to put the speaker's attention on the phonological units that those processes are specialized to manage. Thus, a complex design, functionally completely different from animal communication, must have evolved for spoken language (Pinker 1994, pg. 362).

The phonetic units of speech are the vehicles of every language on earth, and they are commanded by every neurologically normal human being. Lai et. al. (2001) found a gene, FOXP2, which seems to be involved in speech. The regulating gene, located on chromosome 7, was discovered while studying a family most of whose members had troubles controlling their lips and tongue and forming words. More recently, Enard et. al. (2002) studied FOXP2's evolutionary history by comparing versions of the gene in various primates and mice. According to these comparisons FOXP2 has remained essentially unaltered during mammalian evolution, but it changed in humans after the hominid line of descent had split off from the closely related chimpanzee one. The changes in the gene are universal in human populations. Enard et. al. suggest that the changes affected articulation and they estimate that the human version of the gene emerged only 120,000 years ago. Perhaps this mutation of the FOXP2 was the final adjustment that allowed speech to become autonomous, freeing the hands for the development of technologies. Thus, writing and reading did not evolve as part of the language faculty and, therefore, writing and reading differ biologically from speech, being intellectual achievements in a way that speech is not. Many languages do not even have a written form, and, among those that do, some competent speakers find it impossible to master. Awareness of phonological structure is obviously necessary for anyone who would make proper use of an alphabetic script, but such awareness would not normally be a consequence of having learnt to speak.

18.2 Emergence of Symbolic Species

The Victorian people must have been quite shocked when Darwin presented in his Descent of Man (1874) that man evolved from apes. According to Klein's scenario (Klein, 2000), the first primate with bipedal locomotion (Ardipthecus ramidus) lived on African savanna roughly 4.4 million years ago and it took about 2 million years of additional evolution before the first crude tools appeared in the paleontological record about 2.5 million years ago. Brain expansion in homo line begins around 1.2. million years ago and the period of most rapid brain expansion occurred between 500 and 100 thousand years ago. However, all human fossils from 30,000 years ago to today share the same modern anatomical form: a distinct skull shape, a large brain (1,350 cubic centimeters), a chin and a lightly built skeleton. Neanderthals were as human as we are but something dramatic must have happened about 30,000 years ago when Neanderthals suddenly went extinct.

Neanderthals' disappearance coincided with the arrival of the anatomically modern Homo sapiens (Homo sapiens sapiens). Genetic evidence reveals that Neanderthal DNA is distinct from that of modern humans, and it implies that the two lineages diverged perhaps 400,000 years ago. Archeological artefacts left behind show that 100,000 years ago Neanderthals and Homo sapiens were quite similar culturally. However, about 40,000 to 50,000 years ago, a massive transformation occurred (Johanson, 2001; Klein, 2000). Tools became diverse and tailored for different purposes, burials became elaborate and hunters began to target dangerous large animals. This "creative explosion" was almost exclusively limited to Homo sapiens.

Deacon (1997) suggests that symbolic communication originating from new brain adaptations in Homo sapiens sapiens made possible better cultural information transmission from one generation to another and hence better organizational skills that permitted more efficient utilization of sources. Thus, a modern man was equipped with neural prerequisities for the use of symbols in communication, while the Neanderthals were evolved differently in this respect. Neither non-human primates seem to have this adaptation. According to an alternative explanation, there is no specific adaptation for the symbolization per se but adaptation was for understanding others on analogy with the self and symbols then developed as a kind of natural consequence (Chomsky, 1991; Tomasello, 2003). From the comparative perspective, probably the potential for symbolism exists in any animal with a brain of sufficient complexity.

Studdert-Kennedy and Goldstein (2003) suggest that once gestures of distinct organs had evolved as discrete, combinable units, expansion of the phonological systems have occurred by sociocultural processes without any further genetic change. On this view, speech as a motor function evolved

from phylogenetically ancient mammalian oral capacities for sucking, licking, swallowing and chewing (MacNeilage, 1998). For example, sucking, licking, and tongue actions for swallowing might have initiated neuroanatomical differentiation of the mammalian tongue, which the evolution of speech carried further by differentiating tongue tip, tongue body, and tongue root into independent organs of phonetic action. On this view, the basic unit of phonological structure is gesture, not the sounds those gestures produce.

It may be that that the human brain and body were at time 'language-ready' in the sense that the first Homo sapiens used a form of vocal communication which was but a pale approximation of the richness of language as we know it today. The Mirror System Hypothesis (Arbib, 2003) suggests that the functional specialization of human Broca's area derives from an ancient mechanism related to the production and understanding of motor acts. The mirror system's capacity to generate and recognize a set of actions provides the evolutionary basis for language parity, in which an utterance means roughly the same for both speaker and hearer. Therefore, Arbib (2003, pg. 194) states:

> extension of the mirror system from a system for recognition of single actions to a system for recognition and imitation of compound actions was one of the key innovations in the brains of hominids relevant to language.

According to this motor theory of speech (for a review, see e.g. Liberman 1996) the gestures are specifically phonetic, having evolved solely for the purpose of phonological communication. Therefore, apprehending phonetic structures has to be managed by a distinct, language-specific system that has its own phonetic domain and its own phonetic mode of processing served by a neurobiology of its own (speech module). The motor theorists suggest that the biology of language incorporates a precognitive specialization for the production and perception of vowels and consonants, and that perception of those is therefore immediate; there is no translation from a nonphonetic (auditory) representation because there is no such representation for speech.Thus, early hominids changed by adopting for communicative use an apparatus already divided into discrete units and specialized perceptual system for the recognition of articulatory gestures from the continuously varying acoustic signals (pre- or protolanguage). These adaptations in production and perception of speech finally resulted in symbolic communication by exhaled breath (language). Thus, in addition to changes in the organization of the brain, more peripheral adaptations were also needed for spoken language to be favoured in natural selection.

During the evolution of language speaking and listening became so tightly integrated that they seem to be merely two different manifestations of a single linguistic faculty rather than two separate abilities, coordinate but distinct. The distinction between speaking and listening is clear at the pheripheral

level, because they are based on fundamentally different organs. At a more central level, however, the distinction is less clear; speaking and listening simultaneously would not be so difficult, if they were not integrated at some point. Therefore, the processes of production and perception must somehow be linked and, consequently, their sensorimotor representations must, at some point, be the same. The evolution of brain guaranteed this parity (or the 'mirror' property) between speaking and listening and, thus, speech signals became more relevant for man than other acoustic signals in the environment and became linked to units of language (Liberman, 1996).

18.3 Language and the Brain

Inhuman rationalist principles in the philosophy of science have held on for centuries also in the study of language evolution. Accordingly, it has been a tacit assumption in linguistics and psychology that the purely physical or biological aspects of language should be distinguished from the psychological aspect, and that only the latter belongs to the study of language (Chomsky, 1965). Nevertheless, human language is primarily spoken, which suggests that its evolution must have been constrained by the speech apparatus and the auditory system. In recent years alternative views based on this perspective have emerged, indicating that rather than being two independent domains, the physical and psychological aspects overlap significantly (Diehl, 1991). Therefore, theories of language must link up with theories of brain function. Otherwise the study of language degenerates into a signal-processing oriented or a formalist discipline, both perfectly possible per se but remote from the study of what actually takes place in human beings when something is articulated or perceived.

From the evolutionary perspective, the brain was not built like a computer with a special design in mind but natural selection is responsible for its development. In this process of millions of years of evolution, new anatomical structures and functions developed in succession in relatively distinct stages from existing structures (Lamendella, 1980). These changes often involved increases in the anatomical size and configuration of particular structures, qualitative changes in physiological and functional organization, and increases in the overall information processing potential as existing structures took new functions. New structures arose and carried out old functions in new ways. Consequently, all parts of the brain are functionally integrated so intimately in the course of evolution that physically distinct neural movements for spoken language cannot be shown on the basis of the gross anatomy of the human central nervous system. Therefore, there is no single site for language in the brain but it is scattered all over the distinct parts of the brain.

Biologically human language originates from earlier pre-adaptations

which pave the way for subsequent adaptive changes (Hurford, 2003). For example, bipedalism set in train anatomical changes which culminated in the human vocal tract. Similarly, changes in human mental capacities were necessary before modern man became ready for language. These cognitive pre-adaptations set forward another process of evolution which led to the appearance of syntax relatively late in the history of man. Syntax involves the stringing together of independent subunits into a larger signal. In phonological syntax in units, like the speech sounds, there is no independent meaning, while in lexical syntax in the units, such as the words, there are meanings which contribute the overall meaning of the whole signal.

It is nowadays commonly accepted that language somehow emerges gradually from highly complex neuronal events which are firmly organized on a time basis. These neuronal events can be referred to as a kind of programme to emphasize the computational character of the higher-level brain functions. The term "serial action programme" (Ingvar, 1983) has been used in neurophysiology to refer to conceptual structures, which is a term used in linguistic literature for temporally organized neuronal events pertaining to language. According to Chomsky (2004), uniquely human component of the language faculty is syntax, varying little among humans and without significant analogue elsewhere. Thus, language is biologically isolated in its essential properties, and a rather recent development in human evolution. Chomsky (1991) has argued that language is not an adaptation at all, but rather is a by-product or side effect of the tremendous growth of the human brain. His argument is that after the brain attained its current size and complexity, language simply emerged spontaneously as one of many side effects. Despite arguing that language is not a designed adaptation produced by evolution, Chomsky nevertheless has argued that the deep structure of the grammar is innate rather than acquired, and universal in all humans.

18.4 Comparative approach to evolution of symbols and syntax

The faculty of language refers to the narrow syntax "machine" (faculty of language in the narrow sense, FLN), which is a computational system operating on syntactic symbols according to specific rules of computation, and generates an infinite number of utterances from a finite set of syntactic symbols (Hauser, Chomsky & Fitch, 2002). FLN represents a "language organ" per se, which is a subsystem of a more complex structure consisting of two interfaces: the Articulatory-Perceptual and the Conceptual-Intentional (faculty of language in the broad sense, FLB). The syntax "machine" was not instantaneously inserted into a mind/brain with the rest of its architecture fully intact. Rather, it is embedded within the broader architecture of the mind/brain and it interacts with other systems. Therefore, the systems within which the lan-

guage faculty is embedded must be able to communicate the expressions of the language and use them as guidelines for thought and action. Similarly, the sensorymotor systems have to be able to read their instructions having to do with sound and the articulatory and the perceptual systems have specific design that enables them to interpret certain properties, and not others. Thus, the focus in explanations of the language faculty shifted from the study of its subcomponents to their interrelations. Hauser and Fitch (2003) suggest that animals lack the capacity of recursion implying that FLN is an adaption produced by evolution, while subsystems that mediate speech production and perception are not. Many characteristics of speech production and perception are also present either in our closest living relatives or in other, more distantly related species.

Many bird species can learn songs with phonological syntax and apes are known to show a pre-syntactic capacity to organize longer sequences of sounds. Thus, it may be that combinatory principles underlying phonology and syntax of human language emerged gradually by a gradually enlarging brain providing more available neurons and more specialized connections between neurons, not greater intelligence per se (Chomsky, 1991; Bickerton, 2003). As a result of an enlarging brain, the modular and highly domain-specific system of recursion may have become penetrable and domain-general, because human mind cannot consist solely of isolated mechanisms that are completely walled off from each other. Selection favors functionally specialized mechanisms that work well together in various combinations and permutations (Buss, 2004). If recursion evolved to solve computational problems such as navigation, number quantifications, or social relationships, then it is possible that other animals have such abilities (Hauser, Chomsky & Fitch, 2002). Under these circumstances, FLN evolved as a by-product of evolution without any survival value.

There are two features of languages, in whatever modality they are expressed, that are generally not present among the communications of other animals: symbols and syntax. Symbolic communication arose first being within the reach of a number of non-human animals, while syntax emerged later remaining beyond the reach of any other species. Thus, protolanguage, with symbolic content but no syntactical structure evolved from different genetic and neural substrate than the subsequent language with syntax (Bickerton, 1995; Pinker, 1994). Okanoya (2003) studied complex vocalizations of Bengalese finches and suggests that Bengalese finches and humans follow similar developmental path. In both species, phonological development precedes syntactical development. Bengalese finches show syntactical control of singing, which may have evolved through the process of sexual selection. Thus, the rudimentary syntax might have evolved also in humans as a by-product of sexual selection without the need for survival value. In addition, there is also

some evidence that complex syntactic rules emerge from quite simple systems of networks, which have a very small number of initial assumptions and learn from imperfect inputs (Tonkes & Wiles, 2003).

18.5 Concluding Remarks

Computers can be programmed for various purposes; in this sense the computer is a domain-general information processor. The idea that there might be some information-processing problems that the human mind was specially designed to process was missing from the cognitive revolution in psychology. For example, the information processing view on speech perception sees the perception of speech as a wholly unexceptional example of the workings of an auditory modality that deals with speech as it does with all other sounds to which the ear is sensitive. In so doing, however, this view sacrifices a more important kind of generality, since it makes speech perception a mere adjunct to language, having a connection to it no less arbitrary than that which characterizes the relation of language to the visually perceived shapes of an alphabet. However, speech is special, but neither more or less so than any other biologically coherent adaptations, including language itself. Thus, the specializations for phonetic and syntactic perception have in common that their products are deeply linguistic, and are arrived at by procedures that are similarly synthetic.

References

Arbib, M. A. 2003. The Evolving Mirror System: A Neural Basis for Language Readiness. In M. Christiansen and S. Kirby, eds., *Language Evolution*, pages 182–200. New York: Oxford University Press.

Bickerton, D. 1995. *Language and Human Behavior*. Seattle: University of Washington Press.

Buss, D. 1933. *Evolutionary Psychology: The New Science of Mind*. New York: Pearson.

Chomsky, N. 1965. *Aspects of the Theory of Syntax*. Cambridge, Mass.: Cambridge University Press.

Chomsky, N. 1991. Linguistics and cognitive science: Problems and mysteries. In A. Kasher, ed., *The Chomskyan Turn*, pages 26–53. Cambridge, MA.: Blackwell.

Chomsky, N. 2004. Language and Mind: Current thoughts and ancient problems. In L. Jenkins, ed., *Variation and Universals in Biolinguistics*, pages 379–405. Amsterdam: Elsevier.

Darwin, C. 1874. *The Descent of Man*. New York: Hurst & Co.

Deacon, T. W. 1997. *The Symbolic Species. The Co-evolution of Language and the Brain*. New York: W. W. Norton & Company.

Diehl, R. L. 1991. The role of phonetics in the study of language. *Phonetica* 48:120–134.

Enard, et. al., W. 2002. Molecular evolution of FOXP2, a gene involved in speech and language. *Nature* 418:869–872.

Hauser, M., N. Chomsky, and T. Fitch. 2002. The faculty of language: what is it, who has it, and how did it evolve. *Science* 298:1569–1579.

Hauser, M. and W. Fitch. 2003. What are the Uniquely Human Components of the Language Faculty? In M. Christiansen and S. Kirby, eds., *Language Evolution*, pages 158–181. New York: Oxford University Press.

Hurford, J. R. 2003. The Language Mosaic and its Evolution. In M. Christiansen and S. Kirby, eds., *Language Evolution*, pages 38–57. New York: Oxford University Press.

Ingvar, D. H. 1983. Serial aspects of language and speech related to prefrontal activity: A selective review. *Human Neurobiology* 2:177–189.

Johanson, D. 2001. Origins of modern humans: Multiregional or out of Africa? *www.actionbiosciences.org* .

Klein, R. G. 2000. Archeology and the evolution of human behavior. *Evolutionary Anthropology* 9:17–36.

Lai *et. al.*, C. S. L. 2001. A forkhead-domain gene is mutated in a severe speech and language disorder. *Nature* 413:519–523.

Lamendella, J. T. 1980. Neurofunctional Foundations of Symbolic Communication. In M. L. F. . S. H. Brandes, ed., *Symbol as Sense*, pages 147–174. New York: Academic Press.

Liberman, A. 1996. *Speech: A Special Code*. London: The MIT Press.

MacNeilage, P. F. 1998. The frame/content theory of evolution of speech production. *Behavioral and Brain Sciences* 21:499–546.

Okanoya, K. 2003. Sexual display as a syntactical vehicle: The evolution of syntax in birdsong and human language through sexual selection. In A. Wray, ed., *The Transition to Language*, pages 46–63. New York: Oxford University Press.

Pinker, S. 1994. *The Language Instinct*. Middlesex, England: Penguin Press.

Stromswold, K. 1996. The Cognitive and Neural Basis of Language Acquisation. In M. S. Cassaniga, ed., *The Cognitive Neurosciences*, pages 855–870. Cambridge, Mass.: The MIT Press.

Studdert-Kennedy, M. and Goldstein L. 2003. Launching Language. In M. Christiansen and S. Kirby, eds., *Language Evolution*, pages 235–254. New York: Oxford University Press.

Tomasello, M. 2003. On the Different Origins of Symbols and Grammar. In M. Christiansen and S. Kirby, eds., *Language Evolution*, pages 94–110. New York: Oxford University Press.

Tonkes, B. and J. Wiles. 2003. Methodological issues in simulating the emergence of language. In A. Wray, ed., *The Transition to Language*, pages 226–251. New York: Oxford University Press.

19

Data Mining Meets Collocations Discovery

HELENA AHONEN-MYKA AND ANTOINE DOUCET

In this paper we discuss the problem of discovering interesting word sequences in the light of two traditions: *sequential pattern mining* (from data mining) and *collocations discovery* (from computational linguistics). Smadja (1993) defines a *collocation* as "a recurrent combination of words that co-occur more often than chance and that correspond to arbitrary word usages." The notion of arbitrariness underlines the fact that if one word of a collocation is substituted by a synonym, the resulting phrase may become peculiar or even incorrect. For instance, "strong tea" cannot be replaced with "powerful tea". Acquisition of collocations, a.k.a *multi-word units*, are crucial for many fields, like lexicography, machine translation, foreign language learning, and information retrieval. We attempt to describe the collocations discovery problem as a general problem of discovering interesting sequences in text. Moreover, we give a survey of common approaches from both collocations discovery and data mining and propose new avenues for fruitful combination of these approaches.

19.1 Representation of Text and Interesting Sequences

In this section, we discuss several alternatives for representing text and sequences. Moreover, we define the problem of discovering interesting sequences in text.

Assume a text is split into *a set of text fragments*, e.g., into a set of sentences, paragraphs, or documents. Each fragment is a *sequence*, i.e., an ordered list, of *events*. An event is a non-empty unordered set of *items*. Each

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

item belongs to an *alphabet*. If we are interested in word sequences, each event contains one item only, i.e., a word. Events can, however, also have some inner structure. For instance, we could record for each word token its baseform, part of speech, and some morphological information dependent on the part of speech, like in the following example.

i	i	nom	pron
saw	see	past	v
a	a	sg	det
red	red	abs	a
ball	ball	nom	n
and	and	nil	cc
a	a	sg	det
green	green	abs	a
ball	ball	nom	n

In the sample, the first item of an event is an inflected word, the second is the base form, and the fourth is the part of speech. The third item varies based on the part of speech of the word. For instance, 'nom' means nominative (nouns, pronouns), 'abs' an absolute (adjectives; as opposite to comparatives and superlatives), and 'past' means past tense (verbs). Some parts of speech (adverbials, determiners) do not have any specific information. Thus, the third item has a value 'nil' for them.

The *length* of a sequence can be defined as the number of items or as the number of events. Hence, using the first definition, the length of a sequence $\langle sg, det\rangle \rightarrow \langle a\rangle \rightarrow \langle ball, nom, n\rangle$ is six, whereas using the second definition, the length would be three.

In data mining, the sequential pattern discovery problem is usually stated as "*Find all interesting subsequences in the input data.*" If each event contains one item only, the subsequence relation can be defined in the following way.

Definition 1 *A sequence $p = a_1 \cdots a_k$ is a* subsequence *of a sequence q if all the items $a_i, 1 \leq i \leq k$, occur in q and they occur in the same order as in p. If a sequence p is a subsequence of a sequence q, we also say that p* occurs *in q and that q is a* supersequence *of p.*

For instance, the sequence $\langle$*unfair practices*$\rangle$ can be found in all of the three sentences in Figure 1. If an event can contain a set of items, it is enough if some of the items occur in the corresponding event in a longer sequence. For instance, $\langle det\rangle \rightarrow \langle nom, n\rangle$ is a subsequence of $\langle sg, det\rangle \rightarrow \langle a\rangle \rightarrow \langle ball, nom, n\rangle$, but $\langle saw, see, past\rangle \rightarrow \langle v\rangle$ is not a subsequence of $\langle saw, see, past, v\rangle$.

The *interestingness* of a subsequence is usually defined with respect to a set of *constraints*, which are assumed to represent some natural restrictions in

1. The **Congress** subcommittee backed away from mandating specific **retaliation against foreign** countries for **unfair** foreign **trade practices**.
2. He urged **Congress** to reject provisions that would mandate U.S. **retaliation against foreign unfair trade practices**.
3. Washington charged France West Germany the U.K. Spain and the EC Commission with **unfair practices** on behalf of Airbus.

FIGURE 1 A set of sentences (Reuters-21578 1987).

the domain. In practice, the constraints are also used to reduce computational costs. The most common constraint is the *minimum frequency*. The frequency of a (sub)sequence can be, e.g., the number of text fragments that contain it.

Definition 2 *A sequence p is* frequent *in a set of fragments S if p is a subsequence of at least σ fragments of S, where σ is a given frequency threshold.*

If we assume that the frequency threshold is 2, we can find two frequent sequences in our sample set of sentences: $\langle$*congress retaliation against foreign unfair trade practices*$\rangle$ and $\langle$*unfair practices*$\rangle$ (Fig. 1).

As we will see below, the special characteristics of text data usually prohibits discovering all frequent subsequences. Instead, the patterns of interest can be restricted to be *maximal frequent subsequences*.

Definition 3 *A sequence p is a* maximal frequent (sub)sequence *in a set of fragments S if there does not exist any sequence p' in S such that p is a subsequence of p' and p' is frequent in S.*

In our example, the sequence $\langle$*unfair practices*$\rangle$ is not maximal, since it is a subsequence of the sequence $\langle$*congress retaliation against foreign unfair trade practices*$\rangle$, which is also frequent. The latter sequence is maximal.

In addition to a minimum frequency threshold, we can also set a *maximum frequency threshold*. If we prune away the very frequent words, we can reduce the search space significantly. The disadvantage is naturally that we cannot discover any sequences that contain common words, like verb–preposition pairs.

The *internal density* of subsequences can be influenced by constraining the occurrences of events into a predefined window. The size of a window can be a fixed constant, or some natural structure can be taken into account. For instance, the words in a sequence have to occur within a sentence or a paragraph. This latter constraint can be easily implemented by choosing the representation of a text to be a set of sentences or a set of paragraphs, respectively. We can also define a *maximum gap*, which gives the number of other words that are allowed in the text between the words of a sequence. If the maximum gap is zero, we find n-grams in the most common sense, i.e.,

sequences of words, where the words occur consecutively. It is also possible to define a *minimum gap*, although it is harder to find any reasons for that in a general case.

A *minimum* and *maximum length* of sequences can also be defined, although both are problematic in practice. Usually the minimum length of interesting sequences is 2. As the number of frequent sequences decreases radically when the length of sequences increases, we would probably lose a significant part of interesting sequences, if we set the threshold even to 3. The set of frequent pairs naturally also contains a load of uninteresting information, and hence, ignoring them is tempting. It seems, however, to be more reasonable to use some other ways to measure the interestingness than plain length. Setting a maximum length for a sequence may also be problematic, as very long frequent sequences may occur in text. If we set a length threshold to, say, 10, but there is a frequent sequence of length 22 in a text, the discovery process has to output all the 10-subsequences of this long sequence. This would mean outputting thousands of subsequences that actually only represent one sequence. If length is not restricted, the maximal frequent sequences get a chance to be a very compact representation of the regularities in text.

Discovery of interesting sequences in a text is influenced by the special characteristics of textual data. The alphabet size can be 50,000-100,000 words even in a moderate size text collection, which is high compared to alphabets in other application fields, e.g., there are 20 amino acids only. The distribution of words is skewed. There is a small number of words that are very frequent, whereas the majority of words are very infrequent. The words somewhere in the middle, i.e., words with moderate frequency, are usually considered the most interesting and most informative. If the very frequent words are removed, the resulting search space is very sparse. Also the length of the interesting sequences is skewed. There is a large number of short sequences, but also very long sequences are possible. An extreme case is, for instance, if the data set contains several copies of the same document.

19.2 Collocations Discovery

Collocations discovery has been an active research field for decades and various techniques have been explored. Three major types of approaches can be recognized (Schone and Jurafsky 2001): 1) segmentation-based, 2) word-based and knowledge-driven, and 3) word-based and probabilistic. Segmentation-based approaches have focused on identifying words in phonetic streams or in languages that do not include word delimiters. Word-based, knowledge-driven approaches use more or less advanced linguistic filtering, whereas word-based, probabilistic approaches attempt to find collocations using word combination probabilities. Many word-based techniques,

naturally, combine the both approaches.

Choueka et al. (1983) define a collocation simply as "a sequence of adjacent words that frequently appear together". In their approach the sequences are theoretically of any length but were limited to size 6 in practice, due to computational reasons. They experimented on a corpus of 11 million words from the New York Times archive and found thousands of common expressions such as "home run", "fried chicken", and so on. The criteria used to qualify or reject a sequence as a collocation is simply based on a frequency threshold, which makes the results dependent on the size of the document collection.

The technique presented by Mitra et al. (1987) for extracting syntactical phrases is based on a part-of-speech analysis of the document collection. A set of part-of-speech tag sequence patterns are predefined to be recognized as useful phrases. All maximal sequences of words accepted by this grammar form the set of phrases. For instance, a sequence of words tagged as "verb, cardinal number, adjective, adjective, noun" constitutes a phrase of length 5. Every subsequence occurring in this same order is also extracted, with an unlimited gap (e.g., the pair "verb, noun"). This technique defines no minimal frequency threshold.

Church and Hanks defined a collocation to be "a pair of correlated words" (Church and Hanks 1990), that is, as a pair of words that occur together more often than by chance. The correlation is measured by *pointwise mutual information* I:

$$I(x,y) = log_2 \frac{P(x,y)}{P(x)P(y)},$$

where $P(x)$ and $P(y)$ are the probabilities of occurrence of the words x and y, and $P(x,y)$ is the probability that both occur simultaneously. Simultaneous occurrence can be defined in several ways. Two words may occur together, when they are adjacent and in a given order, while a more relaxed definition may require the words to occur within a given window or in the same document in any order.

Building on the work of Choueka et al., Smadja (1993) proposed a hybrid technique that uses statistical measures to find candidate sequences and syntactical analysis to extract collocations. In the statistical phase, the *z-score* of a pair is calculated by computing the average frequency of the words occurring within a 5-word radius of a given word (either forward or backward) and then determining the number of standard deviations above the average frequency for each word pair. Pairs with a *z-score* below a threshold parameter are pruned away. Syntactic filters are then applied to get rid of those pairs which are not considered to be true lexical collocations. For instance, if a candidate is a noun-verb pair, it is accepted only if it is identified either as

a subject-verb or as a verb-object collocation. Following the identification of word pairs, the collocation set is recursively extended to longer sequences, by searching for the words that co-occurred significantly often together with a collocated sequence identified earlier.

A more recent approach of Dias et al. (2000) generates directly all the n-grams, in practice 7-grams, and attempts then to identify their subsequences that are collocations. A sequence is considered to be a collocation, if its words are tighter associated than the words in any of its sub- or supersequences. The association measure used is based on the average expectation of one word occurring in a given position knowing the occurrence of the other $n-1$ words, also constrained by their positions. Also the frequency of a sequence is taken into account. This approach does not need any global thresholds.

Most of the approaches for collocations discovery find rather short sequences only. This is linguistically motivated by experimental evidence that most of the lexical relations associate words separated by at most five other words (Smadja 1993, Dias et al. 2000). As some other languages than English may not share this property, and as some applications, like text summarization, may benefit also from longer interesting sequences, it would be important to have methods that can find longer collocations as well. The approaches described above, however, cannot be straightforwardly extended to find longer sequences, when applied to large corpora. The performance bottleneck can be, at least partially, traced back to processing collocations for each word separately. In the next section we introduce several methods from the data mining community which might solve the problem by processing contexts of many words in parallel, and hence, reducing overlapping work.

19.3 Discovery of Maximal Frequent Sequences

In the spirit of some previous data mining algorithms (Mannila et al. 1995, Srikant and Agrawal 1996), one might suggest the breadth-first, bottom-up approach in Algorithm 1 for the discovery of maximal frequent word sequences. Given the specific characteristics of textual data, the applicability of this algorithm is rather restricted. It generates a lot of candidates and counting of their frequency is slow. In order to answer the question of whether a candidate occurs in an input sequence, all the k-sequences of the input sequence are conceptually generated and compared to the set of candidates. If frequent sequences can be longer than 10 words, this becomes prohibitive.

Some recent methods (Zaki 2001, Tsoukatos and Gunopulos 2001) have proposed ways around this problem. SPADE (Zaki 2001) accelerates frequency counting by using more efficient data structures, but it still enumerates all frequent sequences. DFS_MINE (Tsoukatos and Gunopulos 2001) uses a depth-first approach and, hence, avoids enumerating all subsequences. A

Algorithm 1 *Bottom-up*
Input: a set of sequences (e.g. a set of sentences), a frequency threshold
Output: a set of maximal frequent sequences

1. *Collect all the items of the input sequences, count them, and select the frequent ones.*
2. *Build candidate sequences of length $k+1$ from frequent sequences of length k*
3. *Prune a candidate if some subsequence is infrequent.*
4. *Count the occurrences of the candidate sequences in input and select sequences that are frequent.*
5. *If sequences left: Go to 2.*
6. *Choose the maximal frequent sequences.*

candidate $k+1$-sequence is formed by intersecting a k-sequence with all frequent items. The method has been developed for spatiotemporal data, where the number of different items is much less than in textual data. Intersecting frequent word sequences with all (or many) words is not reasonable.

As we have seen, discovery of maximal frequent sequences in text cannot rely on enumerating all the frequent sequences. Although breadth-first search enables more pruning, it is not feasible, as all subsequences are processed. Depth-first search makes direct computing of maximal frequent sequences possible, but it may have to consider several sequences which are not frequent in the text. We have developed a method MineMFS (Ahonen 1999a,b) that combines breadth-first and depth-first processing. It extracts maximal frequent sequences of any length, i.e., also very long sequences, and it allows an unrestricted gap between words of the sequence. In practice, however, text is usually divided into sentences, which restricts the length of sequences, and gaps as well.

The constraints used in the method are minimum and maximum frequency. Hence, words that are less frequent than a minimum frequency threshold and words that are more frequent than a maximum frequency threshold are first removed. Then, we collect all the ordered pairs, or 2-grams, (A, B) such that words A and B occur in the same sentence in this order and the pair is frequent in the sentence collection.

In the discovery part (Algorithm 2), maximal frequent sequences are discovered directly, expanding each k-sequence that is not a subsequence of any known maximal sequence, until the sequence is not frequent any more. After all k-sequences have been processed, those k-sequences that cannot be used to construct any new maximal sequences are pruned away. The remaining k-

Algorithm 2 *MineMFS.*
Input: G_2*: the frequent pairs*
Output: Max*: the set of maximal frequent sequences*

```
k := 2
Max := ∅
While G_k is not empty
    For all grams g ∈ G_k
        If a gram g is not a subsequence of some m ∈ Max
            If a gram g is frequent
                max := Expand(g)
                Max := Max ∪ max
                If max = g
                    Remove g from G_k
            Else
                Remove g from G_k
    Prune away grams that are not needed any more
    Join the grams of G_k to form G_{k+1}
    k := k + 1
Return Max
```

sequences are joined to form the set of $k+1$-sequences (e.g. AB and BC would produce ABC), and the process is repeated. In our approach the set of k-sequences restricts the depth-first search. Although the alphabet size is large, we have to check only a few alternatives for expanding a sequence. As we do not enumerate all subsequences, we do not have to restrict the length of the sequences.

For experiments the publicly available Reuters-21578 newswire collection (Reuters-21578 1987), which contains about 19,000 non-empty documents, has been used. The average length of the documents is 135 words. Originally, the documents contain 2.5 million words. We have implemented the MineMFS algorithm in Perl and performed experiments using several values for a minimum frequency threshold and a maximum frequency threshold. For instance, with minimum threshold 5 and maximum threshold 200, we found 25,000 frequent 2-sequences, 6,100 3-sequences, 3,700 4-sequences, and so on. In this case, the longest frequent sequences had 20 words. Respectively, with minimum threshold 15 and maximum threshold 500, we found 9,000 2-sequences, 1,600 3-sequences, 650 4-sequences, and finally one sequence of 15 words.

19.4 Discussion

Data mining research often concentrates on developing efficient algorithms with simple frequency-based interestingness measures only. Hence, a new avenue could be to embed interestingness considerations of collocations discovery methods into some data mining algorithms. We have also identified a clear need for more general approaches: both data mining and collocations discovery methods are often developed with some specific data type, task, or language in mind, and, hence, they are not easily applicable to new areas. In the data mining community this need has recently led to the birth of a new research field: *local patterns detection* (Hand et al. (2002)). Its main objective is to develop a theoretical base that would make it possible to identify which methods and interestingness measures are useful for what purposes. In this same spirit we hope, one day, to be able to develop a collocations discovery method that can be easily tuned for any wish of a potential user, let it be common verb–object relations for a foreign language learner, the most significant topical phrases for information retrieval, or domain-specific terms for a machine translation system.

References

Ahonen, Helena. 1999a. Finding all maximal frequent sequences in text. In *ICML99 Workshop, Machine Learning in Text Data Analysis*. Bled, Slovenia.

Ahonen, Helena. 1999b. Knowledge discovery in documents by extracting frequent word sequences. *Library Trends* 48(1):160–181. Special Issue on Knowledge Discovery in Databases.

Choueka, Y., T. Klein, and E. Neuwitz. 1983. Automatic retrieval of frequent idiomatic and collocational expressions in a large corpus. *Journal for Literary and Linguistic computing* 4:34–38.

Church, Kenneth W. and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics* 16(1):22–29.

Dias, Gaël, Sylvie Guilloré, Jean-Claude Bassano, and José Gabriel Pereira Lopes. 2000. Combining linguistics with statistics for multiword term extraction: A fruitful association? In *Proceedings of Recherche d'Informations Assistee par Ordinateur 2000 (RIAO 2000)*.

Hand, David, Niall Adams, and Richard Bolton, eds. 2002. *Pattern Detection and Discovery, ESF Exploratory Workshop, London, UK, September, 2002*. Lecture Notes in Artificial Intelligence. Springer.

Mannila, Heikki, Hannu Toivonen, and A. Inkeri Verkamo. 1995. Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210–215. Montreal, Canada.

Mitra, Mandar, Chris Buckley, Amit Singhal, and Claire Cardie. 1987. An analysis of statistical and syntactic phrases. In *Proceedings of RIAO97, Computer-Assisted Information Searching on the Internet*, pages 200–214.

Reuters-21578. 1987. Text Categorization Test Collection, Distribution 1.0. http://www.daviddlewis.com/resources/testcollections/reuters21578.

Schone, Patrick and Daniel Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In L. Lee and D. Harman, eds., *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 100–108.

Smadja, Frank. 1993. Retrieving collocations from text: Xtract. *Journal of Computational Linguistics* 19:143–177.

Srikant, Ramakrishnan and Rakesh Agrawal. 1996. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology, EDBT'96*, pages 3–17.

Tsoukatos, Ilias and Dimitrios Gunopulos. 2001. Efficient mining of spatiotemporal patterns. In *Proceedings of the SSTD 2001*, no. 2121 in Lecture Notes in Computer Science, pages 425–442.

Zaki, Mohammed J. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning* 42(1):31–60.

20

Semantic Morphology

BJÖRN GAMBÄCK

Semantic Morphology addresses the problem of designing the rules needed for mapping between the semantic lexicon and semantic grammar. The text discusses the relation between semantics, lexicon, and morphology in unification-based grammars and builds on the current trends in Computational Semantics to use underspecification and compositionality. The approach to Semantic Morphology advocated here assumes compositional word formation from (semantic) word roots and affixes that are given their own entries in the semantic lexicon. Different feature usages are then utilized to reach the intended surface word-form matches, with the correct feature settings.

20.1 Introduction

The interaction between morphology and the (syntactic) lexicon on one side and the (syntactic) grammar on the other has been discussed at length in various papers and for various languages. However, the parentheses in the previous sentence point to an almost general restriction: the treatment of language structure has focused mainly on the problems relating morphology to syntax, while little attention has been given to the semantics.

With Semantic Morphology we do not mean the issue of how the actual word-forms are located in the input string, but will take for granted that a module is available to do this work in a unification-based grammar setting, for example such a "lazy" implementation of two-level style morphology (Koskenniemi, 1983) as the one of Carter (1995). Thus in essence, there should be a separation of the task of identifying the input word-form and the task of mapping the lexical feature settings into the grammar, as also argued by Trost and Matiasek (1994).

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

The rest of the text will address the issues of designing and implementing unification-based semantic morphological processing. That is, the morphological rules that execute the mapping between the semantic lexicon and (the rest of!) the semantic grammar — and the way in which features can be used in order to restrict the output to only the desired forms. In doing so, some practical implementations will be discussed, in particular for Japanese and Swedish. Firstly, though, we should note that there have been three strong trends in the Computational Linguistic community during the last decades, both in unification-based grammar approaches in general as well as in most approaches to Computational Semantics:

1. keep as much as possible of the semantic information lexicalized,
2. build complex structures in a compositional manner, and
3. postpone decisions as long as possible.[1]

The first two trends are the topics of the next section, while the third trend is discussed in Section 20.3. Then Section 20.4 introduces some of the work on separating out Semantic Morphology, while Sections 20.5 and 20.6 go into some examples for Japanese and Swedish, respectively. Finally, Section 20.7 sums up the discussion.

20.2 Lexicalization and compositionality

The trend to keep most of the information in the lexicon (rather than in the grammar rules, as traditionally) aims to keep the grammar rules as simple as possible and the number of distinct grammar rules as low as possible — which in turn may result in rather complicated lexica; lexica that are hard, or even impossible, to clearly separate from the grammar proper. On the morphology side, the solution adopted here is the one of introducing affixes as lexical categories, that is, that word formation is given as a compositional addition of affixes to the word roots.

Compositionality may be defined rather strictly so that the interpretation of a phrase always should be the (logical) sum of the interpretations of its subphrases. A semantic formalism being compositional in this strict sense would also trivially be monotonic, since no destructive changes would need to be undertaken while building the interpretation of a phrase from those of its subphrases.[2] In effect then, all the information from the terminal nodes would be passed up to the input (top-level) nodes of the grammar.

[1] A fourth strong trend has been to do away with all "deep" level processing and only use shallow rules or statistical models. However, a discussion of the treatment of morphology in such a "shallow" approach is outside of the scope of this text.

[2] A semantic representation is monotonic if and only if the interpretation of a category on the right side of a rule subsumes the interpretation of the left side of the rule.

However, compositionality is more commonly defined in a wider sense, allowing for other mappings from subphrase-to-phrase interpretation than the sum, as long as the mappings are such that the interpretation of the phrase still is a function of the interpretations of the subphrases. A common such mapping is to let the interpretation of the phrase be the interpretation of its (semantic) head modified by the interpretations of the adjuncts. If this modification is done by proper unification, the monotonicity of the formalism will still be guaranteed.

In general we need morphology and grammar rules for addition of already manifest semantic information (e.g., from the lexicon) and ways of passing non-manifest information (e.g., about complements sought). Assuming a normalised structure, we can then allow for information passing in three ways: trivial composition, function-argument application, and modifier-argument application. The trivial composition manifests itself mainly in rules that are inherently (semantically) unary branching. That is, rules that either are syntactically unary branching, or where the semantics of at most one of the daughter (right-hand side) nodes need to influence the interpretation of the mother (left-hand side) node.

The two types of application rules are quite similar to each other and appear on all (semantically) binary branching rules of the grammar. In both application rule types, the bulk of the semantic information is passed to the mother node from the semantic head among the daughter nodes. However, in functor-argument application the functor is the semantic head, while in modifier-argument application the argument is the semantic head.

The main difference between the two types pertains to the (semantic) subcategorisation schemes: In functor-argument application, the functor subcategorises for the argument, the argument may optionally subcategorise for the functor, and the mother's subcategorisation list is the same as the functor's, minus the argument. Letting **main** intuitively identify the semantic information, **subcat** the subcategorisation list, and *Functor* the semantic head, we get:

(1)
$$\begin{array}{ccccc} \textit{Mother} & & \textit{Functor} & & \textit{Argument} \\ \begin{bmatrix} \textbf{main} & \boxed{1} \\ \textbf{subcat} & \langle \boxed{3} \rangle \end{bmatrix} & \Rightarrow & \begin{bmatrix} \textbf{main} & \boxed{1} \\ \textbf{subcat} & \langle \boxed{2} \mid \boxed{3} \rangle \end{bmatrix} & & \begin{bmatrix} \textbf{main} & \boxed{2} \\ \textbf{subcat} & \langle \boxed{1} \rangle \end{bmatrix} \end{array}$$

In modifier-argument application, the modifier subcategorises for the argument (only), while the argument does not subcategorise for the modifier; its subcategorisation list is passed unchanged to the mother node. This is shown schematically in (2), with *Argument* being the semantic head:

(2)
$$\begin{array}{ccccc} \textit{Mother} & & \textit{Modifier} & & \textit{Argument} \\ \begin{bmatrix} \textbf{main} & \boxed{1} \\ \textbf{subcat} & \langle \boxed{2} \rangle \end{bmatrix} & \Rightarrow & \begin{bmatrix} \textbf{main} & \\ \textbf{subcat} & \langle \boxed{1} \rangle \end{bmatrix} & & \begin{bmatrix} \textbf{main} & \boxed{1} \\ \textbf{subcat} & \langle \boxed{2} \rangle \end{bmatrix} \end{array}$$

20.3 Ambiguity and underspecification

The third trend concerning postponing decisions relates to the problem of ambiguity. Amongst others, ambiguity in a natural language expression may be due to the fact that one of the words used may not have a unique meaning, that more than one syntactic structure may be assigned to the expression, or that the scope relations are not clear. Ambiguities of this kind decrease processing efficiency, since usually all of the possible interpretations have to be assumed to be right until hard facts prove the contrary. The bad news is that this normally happens after a lot of processing has been done.

A way around this dilemma is to have a common representation for all of the possible interpretations of an ambiguous expression, as in the so-called Quasi-Logical Form notation introduced by Alshawi and van Eijck (1989). Following Reyle (1993), the term *underspecification* has been the accepted one to describe this idea. The basic strategy is not to use representations that encode a concrete interpretation but a *set* of interpretations. Thus, the representations are underspecified with respect to one single specific interpretation.

Most work on underspecification has concentrated on scopal ambiguities and anaphora; however, Pinkal (1996) extends the theory of underspecification and discusses several phenomena that lend themselves to this type of compact representation: local ambiguities (e.g., lexical ambiguities, anaphoric or deictic use of pronouns), global ambiguities (e.g., scopal ambiguities, collective-distributive readings), and ambiguous or incoherent non-semantic information (e.g., PP-attachment, number disagreement). Another argument (in addition to the issues related to processing) for underspecified representations is the observation that there is evidence that humans use underspecified information when dealing with natural language. Pinkal (1999) gives a good overview of different approaches to underspecification and also argues at length for its cognitive motivations based on the fact that humans are able to draw inferences from underspecified semantic information.

In order to represent underspecifation, we will assume a semantic representation language such as the ones described by Bos et al. (1996) and Copestake et al. (1999), that is, a language of 'flat' structures which assigns a unique label (name) to every basic formula of the object language with scope (appearing on quantifiers and operators) being represented in an underspecified way by variables ranging over labels. The labeling of conditions is used to make it easier to refer to a particular condition, enabling us to state constraints on the relations between the conditions at the meta-level.

For building these representations we use the operations described above in order to compositionally combine simple representations into complex ones. In addition, a we use a three-place structure referred to as the *context*. It contains the representation's main instance, **inst** (the label of the main event,

which normally is the verb) and two functions that help us keep track of a couple of special labels. These are **main**, the label of the semantic head of the representation, and **top**, the top-most label of the semantic structure.

20.4 Related work

One reason for the lack of interest in computational semantic morphology is that there is a straightforward way to completely ignore it! A common solution is to let the syntactic part of the morphology do all the work and let the semantics "piggyback" on that, letting the semantic lexicon handle the cases where this cannot be done. Accordingly, the German version of the Verbmobil grammar (Bos et al., 1996) let the syntax resolve all inflectional affixing, while verb prefixing (which is rich in German) was fully specified in the lexicon. This means that, e.g., *durchlaufen* (run through) and *durchleben* (live through) need two separate entries in the semantic lexicon, neither of which relate directly to the compositional parts. Thus the "straightforward" solution is possible, but neither elegant nor implementationally attractive. It makes more sense to allow each of the different parts of the word to have their own entries in the semantic lexicon and to apply semantic morphological rules to the parts in order to build the overall semantic interpretation of the word.

There has been some work on relating morphology to semantics within the Lexical Functional Grammar (LFG) and Head-Driven Phrase Structure Grammar (HPSG) traditions. In LFG, Sadler and Nordlinger (2006) argue for treating the problem of case-stacking[3] by connecting the morphology to LFG's functional descriptions in a tree-based fashion. Andrews (2005) argues against this and instead proposes a flat notation. In the HPSG school, most work on semantics has during the last decade concentrated on (flat) Minimal Recursion Semantics, MRS (Copestake et al., 1999). However, these efforts have mainly been devoted to the grammar as such and have more or less disregarded the morphological semantics. The main exceptions to this concern the work on HPSG for Japanese (e.g. Siegel and Bender, 2002).

A recent alternative to MRS is LRS, Lexical Resource Semantics (Sailer, 2004) which aims to separate out the description of local semantic phenomena (such as selectional restrictions and linking, the mapping between semantic roles and syntactic complements) from the non-local (clausal) semantics. In effect, the representation of local semantics in LRS takes the "semantic-head based resolution" of Gambäck and Bos (1998) as a starting point, but extends it and formalises it. Riehemann (1998) argues for an approach in which generalisations from existing words are expressed as schemata, organised in an HPSG-style inheritance network. This is attractive and elegant, although efficiency of an implementation of it still has to be demonstrated.

[3] When a single word contains multiple case markers.

20.5 Japanese morphology

Here we will instead adopt an alternative solution to morphology, where affixes are given specific lexical entries. A very clear example of this kind of treatment can be seen in Japanese. Japanese verbs exhibit affix-based inflectional morphology in its own right, but also more specific phenomena such as the usage of light verbs and particles (especially postpositional) are common. By including the verbal affixes in the semantic lexicon we can treat them and the postpositional particles in a uniform way. Consider as an example the verb phrase *haitte orimasu* in (3).

(3) *itsumo iroiro kaigi ga hait- te ori- masu*
always various meeting NOM be-put-in PART ASP HON+PRES
'all types of meetings are scheduled every day'

Here *hait* is the main verb and *ori* an auxiliary, while *te* and *masu* are inflectional affixes. The core semantical information comes from the main verb, so that the affixes can be treated as modifiers of the respective verb and the auxiliary as a modifier of the main verb. Thus we can, for example, let the lexical entry for *masu* mainly introduce the semantic information for representing the honorific form and pass it up in a purely compositional manner in the morphological analysis tree. The lexical entry for the honorific affix would basically look as (4). So, the only argument which *masu* subcategorises for is its verb, which in turn introduces the discourse marker labeled as $\boxed{1}$.

(4)
$$\begin{bmatrix} \text{RELN} & \texttt{masu} \\ \textit{context} & \begin{bmatrix} \textbf{inst} & \boxed{1} \\ \textbf{main} & \boxed{2} \\ \textbf{top} & \boxed{3} \end{bmatrix} \\ \text{TENSE-ASPECT} & \begin{bmatrix} \textbf{main} & \boxed{2} \\ \text{TENSE} & \text{present} \\ \text{HONORIFIC} & \texttt{masu} \end{bmatrix} \\ \text{MODIFIER} & \text{yes} \\ \text{SUBCAT} & \left\langle \begin{bmatrix} \textit{context} & \begin{bmatrix} \textbf{inst} & \boxed{1} \\ \textbf{main} & \overline{} \\ \textbf{top} & \boxed{3} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix}$$

The most important part of the entry in (4) is the feature-bundle designated TENSE-ASPECT. Here it introduces two things, the present tense and the honorific level which can be viewed as a sort of aspectual information. The honorific level is set simply to `masu` and will in due time be bound to the discourse marker by the **main** label of *masu*, $\boxed{2}$ — thus the lexical entry in effect

introduces a honorific aspect on the main verb. There is no need to refer to the **main** label of the main verb (shown by the '__'), but its **top** label $\boxed{3}$ is bound to the **top** label of *masu*, meaning that the honorific aspect and the present tense will take the same scope as the verb (i.e., normally over the entire sentence). This is not very important for the present discussion, but obviously not a necessary restriction. Bonami (2001) suggests including an (underspecified) scopal restriction in the lexical entry for the tense relation itself, allowing it to take a different scope than the other elements of the tense-aspect structure.

In the same fashion, *ori* would introduce a progressive aspect, while the affix *te* basically would not add anything to the semantics. The verb *hait* is in itself intransitive and thus subcategorises for one argument, the subject. The entire verb phrase structure would then be built recursively using the modifier application rule of Section 20.2. Filling in the schematic rule (2) on Page 206 gives us an overall structure like the one in (5).

(5)
$$\begin{bmatrix}
\text{RELN} & \texttt{hait} \\
\textit{context} & \begin{bmatrix} \textbf{inst} & \boxed{1} \\ \textbf{main} & \boxed{2} \\ \textbf{top} & \boxed{3} \end{bmatrix} \\
\text{TENSE-ASPECT} & \begin{bmatrix} \textbf{main} & \boxed{2} \\ \text{TENSE} & \text{present} \\ \text{ASPECT} & \text{progressive} \\ \text{HONORIFIC} & \texttt{masu} \end{bmatrix} \\
\text{MODIFIER} & \text{no} \\
\text{SUBCAT} & \left\langle \begin{bmatrix} \textit{context} & \begin{bmatrix} \textbf{inst} & \boxed{4} \\ \textbf{main} & __ \\ \textbf{top} & \boxed{3} \end{bmatrix} \end{bmatrix} \right\rangle \\
\text{ROLE} & \texttt{subject(}\boxed{2}\texttt{,}\boxed{4}\texttt{)}
\end{bmatrix}$$

Nicely enough, we would need to make no principal distinction between the applications of the affixes to the verbs and the application of the auxiliary to the main verb. Quite importantly, there would also be no fundamental distinction between the behaviour of these morphology level rules and the rules, for example, for the application of postpositions to NPs to build PPs.

The basic construction in the Japanese syntax is the PP. A PP may be constructed in a range of different ways, the base case, however, being PP $\rightarrow$ NP P. Semantically, the P in this rule is treated uniformly (for all types of postpositions) as a functor applying to the NP, that is, using the functor-argument application rule (1) shown schematically on Page 206.

20.6 Swedish morphology

For an inflectional language like Swedish, where, for example, most of the tense and aspect information can be found in the suffix of the main verb, it is natural to view the tense-aspect information as forming a function of the affix — the information is then filtered up from the verbal affix to the verb phrase. Most work on morphology for Swedish and other Scandinavian languages has concentrated on the purely syntactic side (e.g. Karlsson, 1992; Gambäck, 2001). However, the treatment of non-compositional Danish phrasal verbs in PAROLE/SIMPLE by Pedersen and Nimb (2000) follows the same lines as here by advocating a "split late" strategy where phrasal verbs are singled out as late as possible in the morphological processing, that is, in the semantic part of it.

The lexicon form of choice for Swedish verbs is the imperative, since this form constitutes the stem of most other inflections. For tense and aspect purposes, however, the imperative is a bit peculiar: it stands almost on the side of the entire tense-aspect system. Thus the lexicon contains stems for which the tense-aspect information is only partially instantiated. The (normally) full instantiation is obtained by the inflection in morphology rules as the following schematic one:

(6)

$$\begin{array}{ccccc} \textit{Mother} & & \textit{Verb} & & \textit{Suffix} \\ \begin{bmatrix} \textbf{main} & \boxed{1} \\ \textbf{ten-asp} & \boxed{3} \\ \textbf{subcat} & \langle \boxed{2} \rangle \end{bmatrix} & \Rightarrow & \begin{bmatrix} \textbf{main} & \boxed{1} \\ \textbf{ten-asp} & \overline{} \\ \textbf{subcat} & \langle \boxed{2} \rangle \end{bmatrix} & & \begin{bmatrix} \textbf{main} & \overline{} \\ \textbf{ten-asp} & \boxed{3} \\ \textbf{subcat} & \langle \boxed{1} \rangle \end{bmatrix} \end{array}$$

where the mother verb is formed by adding a suffix to the daughter verb (i.e., the stem form). The tense-aspect information from the suffix is passed up to the inflected verb. This is also the only (semantic) information added by the suffix; the other parts of the mother-verb semantics come from the daughter. An example of a suffix entry is the one in (7) for the ending '-r', which is used to form the present tense when added to the stem of verbs belonging to the first (e.g., *menar*) and third declension (*sker*) as well as those belonging to the third subgroup of the fourth declension (*ser*).

Just like the rules for affixing, we can allow for rules, for example, for the construction of particle verbs simply by including the particle on the (semantic) subcategorisation list of the verb and having a semantic morphology rule for V → P V. Wolters (1997) thus proposes a solution to the German prefix verb problem (Section 20.4) in which each verb's lexical entry contains an indication of which prefixes it may combine with in an HPSG framework. Or rather, which *senses* of the prefixes a verb may combine with in order to form specific interpretations.

(7)
$$\begin{bmatrix} \text{RELN} & \texttt{'-r'} \\ \textit{context} & \begin{bmatrix} \textbf{inst} & \boxed{1} \\ \textbf{main} & \boxed{2} \\ \textbf{top} & \boxed{3} \end{bmatrix} \\ \text{TENSE-ASPECT} & \begin{bmatrix} \textbf{main} & \boxed{2} \\ \text{TENSE} & \text{present} \\ \text{ACTION} & \text{non-progressive} \\ \text{VOICE} & \text{active} \end{bmatrix} \\ \text{MODIFIER} & \text{yes} \\ \text{SUBCAT} & \left\langle \begin{bmatrix} \textit{context} & \begin{bmatrix} \textbf{inst} & \boxed{1} \\ \textbf{main} & \_\_ \\ \textbf{top} & \boxed{3} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix}$$

20.7 Summary

The text has advocated singling out Semantic Morphology as a topic in its own right. This contrasts with many approaches to unification-based grammars where syntax and semantics are treated in parallel, as well as with approaches where the syntax takes total control of the morphology. A key aspect of the treatment presented here is to introduce affixes as their own entries in the semantic lexicon.

Acknowledgements

This work was influenced by the views and efforts of many people: several of my former colleagues and most of the people mentioned in the reference list. Thanks to all of you.

References

Alshawi, Hiyan and Jan van Eijck. 1989. Logical forms in the Core Language Engine. In *Proc. 27th Annual Meeting of the Association for Computational Linguistics*, pages 25–32. ACL, Vancouver, British Columbia.

Andrews, Avery D. 2005. F-structural spellout in LFG morphology. Manuscript. Australian National University, Canberra, Australia.

Bonami, Olivier. 2001. A syntax-semantics interface for tense and aspect in French. In F. van Eynde, L. Hellan, and D. Beermann, eds., *Proc. 8th International Conference on Head-Driven Phrase Structure Grammar*, pages 31–50. Trondheim, Norway: CSLI Publications.

Bos, Johan, Björn Gambäck, Christian Lieske, Yoshiki Mori, Manfred Pinkal, and Karsten Worm. 1996. Compositional semantics in Verbmobil. In *Proc. 16th International Conference on Computational Linguistics*, vol. 1, pages 131–136. ACL, København, Denmark.

Carter, David. 1995. Rapid development of morphological descriptions for full language processing systems. In *Proc. 7th Conference of the European Chapter of the Association for Computational Linguistics*, pages 202–209. ACL, Dublin, Ireland.

Copestake, Ann, Dan Flickinger, Ivan Sag, and Carl Pollard. 1999. Minimal Recursion Semantics: An introduction. Manuscript. CSLI, Stanford, California.

Gambäck, Björn. 2001. Unification-based lexicon and morphology with speculative feature signalling. In A. Gelbukh, ed., *Computational Linguistics and Intelligent Text Processing: Proc. 2nd International Conference*, no. 2004 in Lecture Notes in Computer Science, pages 349–362. Mexico City, Mexico: Springer-Verlag.

Gambäck, Björn and Johan Bos. 1998. Semantic-head based resolution of scopal ambiguities. In *Proc. 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, vol. 1, pages 433–437. ACL, Montreal, Canada.

Karlsson, Fred. 1992. SWETWOL: A comprehensive morphological analyser for Swedish. *Nordic Journal of Linguistics* 15(1):1–45.

Koskenniemi, Kimmo. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Doctor of Philosophy Thesis, University of Helsinki, Dept. of General Linguistics, Helsinki, Finland.

Pedersen, Bolette Sandford and Sanni Nimb. 2000. Semantic encoding of Danish verbs in SIMPLE: Adapting a verb-framed model to a satellite-framed language. In *Proc. 2nd International Conference on Language Resources and Evaluation*, pages 1405–1412. ELRA, Athens, Greece.

Pinkal, Manfred. 1996. Radical underspecification. In P. Dekker and M. Stokhof, eds., *Proc. 10th Amsterdam Colloquium*, vol. 3, pages 587–606. Amsterdam, Holland.

Pinkal, Manfred. 1999. On semantic underspecification. In H. Bunt and E. Thijsse, eds., *Proc. 3rd International Workshop on Computational Semantics*, pages 33–56. Tilburg, Holland.

Reyle, Uwe. 1993. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics* 10:123–179.

Riehemann, Susanne. 1998. Type-based derivational morphology. *Journal of Comparative Germanic Linguistics* 2:49–77.

Sadler, Louisa and Rachel Nordlinger. 2006. Case stacking in realizational morhology. *Linguistics* (to appear).

Sailer, Manfred. 2004. Local semantics in Head-Driven Phrase Structure Grammar. In O. Bonami and P. C. Hofherr, eds., *Empirical Issues in Formal Syntax and Semantics*, vol. 5, pages 197–214. Paris, France: CNRS.

Siegel, Melanie and Emily M. Bender. 2002. Efficient deep processing of Japanese. In *Proc. 3rd Workshop on Asian Language Resources and International Standardization*, pages 31–38. ACL, Taipei, Taiwan.

Trost, Harald and Johannes Matiasek. 1994. Morphology with a null-interface. In *Proc. 15th International Conference on Computational Linguistics*, vol. 1, pages 141–147. ACL, Kyoto, Japan.

Wolters, Maria. 1997. Compositional semantics of German prefix verbs. In *Proc. 35th Annual Meeting and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 525–527. ACL, Madrid, Spain. Student session.

21

Morphological Processing in Mono- and Cross-Lingual Information Retrieval

KALERVO JÄRVELIN AND ARI PIRKOLA

21.1 Introduction

Text-based information retrieval (IR) matches text-based representations of information needs to text-based representations of documents. Since both representations deal with natural language and have different sources and characteristics, their match rarely is perfect. IR therefore has to deal with several difficult problems: First, the request representing the information need often is vague and short thereby providing little evidence for the IR system about desired document features - suggesting document relevance. Secondly, the request wording may be different from that of relevant documents due to many natural language features, e.g., synonymy and inflection. Thirdly, even useless documents may contain many request words.

In this paper we shall focus on text-based representations of documents and requests, and their matching. Our specific focus will be morphological processing of documents and requests in order to derive representations that better support document - request matching. This study is motivated by the morphological variability of natural languages. While much of IR research deals with English, English is morphologically fairly simple. Therefore findings in the English IR context do not necessarily apply IR in other languages with different morphological characteristics. We shall therefore contrast findings in English IR with findings in Finnish IR. Finnish is morphologically much more complex than English. For example, English nouns have singular and plural and two cases while Finnish nouns in principle may have over 2,000 different inflectional forms (Koskenniemi 1983).

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

In IR multiple approaches have been adopted in the handling of morphological variation of words. The baseline is token-based indexing and retrieval - i.e. plain text words are used as such for the representation of documents and requests - with obvious problems when the request word tokens do not match the document word tokens. A simple way to alleviate the problems is to leave the document representation intact, but use a *truncation operation* on the request words to match in the index all document words having the same initial characters. In large text databases truncation tends to match too many words turning queries unmanageably long. Linguistic morphological processing can be alleviated also by approximate string matching, e.g., by *n-gramming* (McNamee and Mayfield 2004).

Among linguistically informed approaches, one possibility is to apply stemming on both request and document words thereby removing much of inflectional variation (Porter 1980). Stemming may however conflate fairly remote words to common stems turning them unspecific or fail to identify the common stem of some words in complex cases. An elaborated approach combining stemming and truncation is *stem generation* (Kettunen et al. 2005). Here several distinct inflectional stems are generated for one lemma before matching the token-based index - yielding shorter and more specific queries. A further possibility consists of the production of all inflectional word forms (Arppe 1996) for request words. However, in morphologically complex languages this tends to lead to excessively long queries. The final approach is *lemmatization* where the lemma of each document and request word is automatically identified and request word lemmas are compared to the lemma-based index.

Lemmatization would be the ideal approach for handling morphology in IR if not for two problems: word form ambiguity and out-of-vocabulary (OOV) words. Words are often ambiguous but may be disambiguated. However, most IR studies on disambiguation have reported no or minor improvements in retrieval performance (Krovetz and Croft 1992; Sanderson 1994). Lemmatizers often cannot handle OOV words (correctly). Often such words are proper names, which tend to be significant words in requests. Their incorrect treatment thus leads to severe loss in IR performance.

In this paper we accept lemmatization as the gold standard for morphological processing in IR and compare the plain words baseline and the morphologically simpler approaches to lemmatization w.r.t. IR performance. A prominent approach in lemmatization is the Two-level Morphology developed by Kimmo Koskenniemi (1983) and implemented in several lemmatization programs for several languages - e.g., FINTWOL, GERTWOL, ENGTWOL, SWETWOL. Riitta Alkula (2000; 2001) conducted the seminal experiments with Finnish morphological processing in IR and the TWOL software (among others). Her studies were performed in the Boolean exact-match re-

trieval environment. In this paper we shall focus however solely on experiments in best-match IR environments.

We shall look at the following research questions:

- Monolingual IR test condition:
  - What is the relative IR performance in Finnish IR of plain words, stemming, and inflectional stem generation w.r.t. lemmatization by FINTWOL.
  - Regarding FINTWOL: what is the relative IR performance of FINTWOL when compounds are split and they are kept intact.
  - What is the relative IR performance in English IR of plain words and stemming w.r.t. lemmatization by ENGTWOL.
- Cross-lingual IR test condition, keeping English as a source language, Finnish, German, and Swedish as target languages:
  - What is the relative retrieval performance in cross-language IR of stemming w.r.t. lemmatization by TWOL.
  - Regarding TWOL: what is the relative IR performance of TWOL when compounds are split and they are kept intact.

We shall review recent empirical findings produced at the University of Tampere (Airio 2005; Kettunen et al. 2005; Kettunen 2005).

This paper is organized as follows. Chapter 2 first discusses morphological differences between Finnish and English, then presents findings on morphological normalization in IR and then discusses morphological processing in cross-language IR. Chapter 3 presents the test settings and Chapter 4 the results. Chapter 5 contains discussion and conclusions.

21.2 Morphological Processing in IR

21.2.1 Morphological Differences between Finnish and English

Morphology studies word structure and formation and consists of inflectional morphology and derivational morphology. The former focuses on the formation of inflectional forms from lexemes. The latter is concerned with the derivation of new words from other words or roots. English and Chinese have a simple morphology whereas many other languages, e.g., Germanic languages or such languages as Finnish are morphologically more complex.

The *Finnish language* is a very inflectional and compound rich language. If Finnish text words are stored in their inflected forms in the database index, this results in clearly greater space requirements for Finnish text compared to that of English texts of corresponding length. For example, Finnish has more case endings than is usual in Indo-European languages. Finnish case endings correspond to prepositions or postpositions in other languages (cf. Finnish *auto/ssa, auto/sta, auto/on, auto/lla* and English in the car, out of the car, into

the car, by car). There are 15 cases, while English has only two (Karlsson 1987).

In Finnish, several layers of endings may be affixed to word stems, indicating number, case, possession, modality, tense, person, and other morphological characteristics. This results in an enormous number of possible distinct word forms: a noun may have some 2,000 forms, an adjective 6,000, and a verb 12,000 forms. Moreover, these figures do not include the effect of derivation, which increases the figures roughly by a factor of 10 (Koskenniemi 1985). Consonant gradation makes the inflection even more complicated, as the stem of a word may alter when certain types of endings are attached to it. For example, the word *laki* (law) has in practice four inflected stems: *laki-*, *lake-*, *lai-*, and *lae-*. The common root of the stems consists of only two characters, which renders it inappropriate as a search key.

Several languages, Germanic and Finno-Ugrian languages included, are rich in compounds in contrast to English, which is phrase-oriented. For example, in Finnish, The Dictionary of Modern Standard Finnish contains some 200,000 entries, of which two-thirds are compound words (Koskenniemi 1983). For example the English phrase 'Turnover Tax Bureau' is *liike|vaihto|vero|toimisto* in Finnish (word boundaries here marked by '|'). In Finnish, compounding results in a problem of retrieving the second or later elements of compounds, for example *verotoimisto* (tax bureau), if the searcher is not able to recall all possible first components.

The fairly simple morphology of English suggests that the costs of morphological processing in IR are low. One may dispense with the morphological processing and still achieve good results. However, stemming has been shown to be useful in English IR (Section 2.2). In contrast to English, the complex morphology of Finnish suggests that simple morphological methods may not be sufficient, but lemmatization or some other sophisticated method is required to achieve the best possible results.

21.2.2 Previous Research

Stemming has been the most widely applied morphological technique in IR. With stemming, the searcher does not need to worry about the correct truncation point of search keys. Stemming also reduces the total number of distinct index entries. Further, stemming causes query expansion by bringing word variants, derivations included, together (see, e.g. Alkula 2001; Krovetz 1993; Pirkola 2001). Some early research results with English collections questioned the effectiveness of stemming (Harman 1991). Later results by, e.g. Krovetz (1993) and Hull (1996) found stemming useful especially when long enough retrieved sets of documents are analyzed. Hull also found out that stemming is always useful with short queries. With short queries and short documents, a derivational stemmer is most useful, but with longer

ones the derivational stemmer brings in more non-relevant documents. Stemming increases search key ambiguity and greedy stemming may be counterproductive. With long queries and documents, relevant material can be identified with conservative stemming. In languages other than English, stemmers have been even more successful than in English text retrieval - e.g., in Slovenian (Popovic and Willett 1992), French (Savoy 1999), Modern Greek (Kalamboukis 1995), and Arabic (Abu-Salem et al. 1999).

The benefits of *lemmatization* are the same as in stemming. In addition, when basic word forms are used, the searcher may match an exact search key to an exact index key. Such accuracy is not possible with truncated, ambiguous stems. Homographic word forms cause ambiguity (and precision) problems - this may also occur with inflectional word forms (Alkula 2001). Another problem is owing to words that cannot be lemmatized, e.g. foreign proper names, because the lemmatizer's dictionary does not contain them. Such problem words need special handling.

Compound words may be split into their components in lemmatization. When indexing a text collection, both compounds and their components may be recorded in the database index thus enabling retrieval through all combinations of compound components. Recent findings suggest that lemmatization with compound splitting improves retrieval performance in Boolean (Alkula 2001) and best-match retrieval (Kunttu 2003). Their most important effects, however, may be the cognitive simplification of query formulation. The searcher is greatly relieved if she need not consider potential expressions like "Verkehrswegeplanungsbeschleunigungsgesetzveränderungsentwurf"[1] when interested in legislation on road planning.

21.2.3 Morphological Processing in Cross-Language Retrieval

Cross-language information retrieval (CLIR) refers to an information retrieval task where the language of queries is other than that of the retrieved documents. Different approaches to cross-language retrieval are discussed in Oard and Diekema (1998). In *dictionary-based CLIR* a standard method is to replace each source language key by all of its target language equivalents included in a translation dictionary (Pirkola 1998; Pirkola et al. 2001). The main problems associated with dictionary-based CLIR are (1) OOV words, (2) morphological processing of keys, (3) phrase identification and translation, and (4) lexical ambiguity in source and target languages. Here our focus is on the problem (2).

Morphological processing is needed in three situations in dictionary- based CLIR: processing of source language search keys for dictionary look- up, processing of inflected dictionary output words, and processing of database

[1] In German - a proposal for changing the law on speeding up the planning of roads - here no compounds.

index keys. Lemmatization is often used in the first stage to facilitate matching of source keys with dictionary headwords (in base forms) also in the case of inflected search keys. Alternatively, source keys and headwords can be conflated into the same form by a stemmer (Davis and Ogden 1997). One problem related to stemming is that different headwords may be conflated into the same form. In our experiments source language (English) keys were lemmatized by ENGTWOL for dictionary look-up (Section 3).

If index keys are stemmed, dictionary output words also have to be stemmed (Davis and Ogden 1997). In the case of the lemmatized index keys, the lemmatization of the output words does not seem necessary, but might be useful since some dictionary output words may be in inflected forms, e.g., some phrase component words (Hedlund et al. 2001).

Regarding word inflection CLIR effectiveness depends to a great extent on the morphological processing of index keys. This issue is the focus of our cross-language IR experiments. We explore the matching of target language queries against different types of indexes as described in Section 3.

21.3 Test Data and Settings

The tests of this study were conducted in the Information Retrieval Laboratory of the Department of Information Studies, University of Tampere. Actual searches were conducted with a probabilistic partial match system, InQuery, version 3.1 (Callan et al. 1992, Broglio et al. 1995) in two different testing environments called *Environment One* and *Environment Two*. In Environment One we studied monolingual Finnish IR, and in Environment Two monolingual Finnish and English IR and cross-lingual English to Finnish, German, and Swedish IR. Next we describe the two environments.

The test collection of *Environment One*, TUTK, contains a full text database of newspaper articles published in three Finnish newspapers in 1988 - 1992 (Sormunen 2000). The database consists of 53,893 articles. The articles of the database are fairly short on average. Typical text paragraphs are two or three sentences in length. The topic set consists of 30 topics. Topics are long: the mean length of the original topics is 17.4 words. The relevance of documents is assessed on a four-level scale. In this study we used a binary relevance scale and combined the documents on the levels 2 and 3 into a class of relevant documents, and the documents on the levels 0 and 1 into a class of irrelevant documents.

We used the following morphological programs: FINTWOL (for lemmatization), MaxStemma (for stem generation), and Finnish Snowball stemmer which is freely available on the Web (http://snowball.tartarus.org). MaxStemma was implemented by Kimmo Kettunen in early 1990's. Its original version is described in more detail in (Kettunen 1991a, 1991b).

The MaxStemma stem generator works in the following fashion: given the base word form (nominative singular for nouns), it produces all the differing inflectional stems of the words. Depending on the input noun, 1 - 5 different stems (including the base form) are produced for a noun. For instance, if the input word is *kissa* ('cat'), the program would generate the following inflectional stems for the word: *kissa, kissoi, kissoj*.

The Snowball stemmer returns stems out of inflected word forms. Snowball is a Lovins' style stemmer that strips off suffixes from the input word according to a suffix list and set of rules and returns stems for the words (Frakes 1992, Porter 2001).

The experiments conducted in *Environment Two* used CLEF (Cross Language Evaluation Forum; http://clef.isti.cnr.it/) data and the UTACLIR query translation system of the University of Tampere. We used CLEF 2003 Finnish, German, Swedish, and English test collections, test topics and relevance assessments. There are 60 CLEF 2003 topics, translated into all the CLEF languages, including the present test languages.

The UTACLIR system utilizes several external language resources (translation dictionaries, stemmers and lemmatizers, and stop word lists) in processing queries for retrieval (Airio et al. 2003). Word processing in UTACLIR proceeds as follows. First the topic words are lemmatized. The existence of a lemmatizer for the source language is vital, because stemmed words are not translatable. The lemmatizer produces one or more basic forms for a token. After normalization, stop-words are removed, and non-stop words are translated. If translation equivalents are found, they are normalized utilizing a lemmatizer or a stemmer, depending on the target index. If translation equivalents are not found, they are identified in the target index by n-gramming the source word. Queries are structured utilizing InQuery's synonym operator: the target words derived from the same source word are grouped into the same synonym group (Pirkola 1998).

For comparing performance of different word normalization tools and decompounding in monolingual and cross-lingual IR different kinds of indexes were created (inflected, stemmed, lemmatized with decompounding, and lemmatized without decompounding). As normalization tools we used TWOLs and Snowball stemmers for Finnish, German, Swedish, and English. Altogether 16 test runs were performed, out of which 7 were monolingual and 9 cross- lingual.

The approach in the *monolingual stemmed runs* was to stem the topic words, and perform retrieval in the stemmed index. In the *monolingual lemmatized runs*, the topic words were lemmatized, and retrieval was performed in the lemmatized indexes. For Finnish there were two lemmatized indexes (compounds were and were not split) and for English one (compounds were not split). In the *inflected word form* runs, topic words were added as such

into the query, and retrieval was performed in the inflected word form index.

For the cross-language IR tests two lemmatized runs (one in the decompounded index and one in the index without decompounding) and one stemmed run were performed for all the language pairs.

21.4 Results

The results of the monolingual Finnish IR experiments in Environment One are presented in Table 1. The results of the monolingual Finnish and English IR experiments conducted in Environment Two are presented in Table 2. The results of the cross-lingual IR experiments are shown in Table 3.

From Table 1 one may see that FINTWOL (lemmatization) performs slightly better than MaxStemma (stem generation). The performance of Snowball (stemming) is clearly below the former. The worst performance was achieved for Plain Words. On the average Plain Words achieved 54.0 % of FINTWOL's performance.

In Environment Two for Finnish monolingual runs the best result was achieved with the decompounded lemmatized index, the next best with the stemmed index, and the worst with the inflected index (Table 2). The results of English monolingual runs are in line with the majority of the earlier results: no statistically significant differences could be found between the inflected run and the normalized runs.

Table 3 shows average precision for the cross-lingual runs. Retrieval in the lemmatized indexes where compounds were split performed best in all the cross-lingual runs. In English-Finnish and English-German, the next best was the run in the lemmatized index without decompounding, and the stemmed run achieved the worst result. In English-Finnish, the stemmed run performed clearly worse than both of the lemmatized runs: the result was 41.4 % worse than that of the run in the lemmatized decompounded index.

In English-Swedish and in English-German, the differences between the two lemmatized runs were statistically significant by the Wilcoxon signed ranks test at the 0.01 level, but differences between the run in the lemmatized index without decompounding and stemmed run were not significant. In English-Finnish the situation is opposite: the differences between the two lemmatized runs were not statistically significant, but between the run in the lemmatized index without decompounding and stemmed run they were significant. All the differences between the cross-lingual stemmed runs and the runs in the lemmatized decompounded indexes were statistically significant at the 0.01 level.

Table 1. The performance of monolingual Finnish runs in Environment One

Morphological tool	Average Precision %	Change % w.r.t FINTWOL
1. FINTWOL	35.0	
2. MaxStemma	34.2	-2.3
3. Snowball	27.7	-20.9
4. Plain Words	18.9	-46.0

Table 2. The performance of monolingual Finnish and English runs in Environment Two

Language	Index type	Average precision %	Change % w.r.t 1a or 2a
1a. Finnish	Lemmatized, split	50.5	
1b. Finnish	Lemmatized, no split	47.0	-7.0
1c. Finnish	Stemmed	48.5	-4.0
1d. Finnish	Inflected	31.0	-38.6
2a. English	Lemmatized, no split	45.6	
2b. English	Stemmed	46.3	+1.5
2c. English	Inflected	43.4	-4.8

Table 3. The performance of cross-language runs with English as the source language

Target Language	Index type	Average precision %	Change % w.r.t 1a, 2a or 3a
1a. Finnish	Lemmatized, split	35.5	
1b. Finnish	Lemmatized, no split	29.0	-18.3
1c. Finnish	Stemmed	20.8	-41.4
2a. Swedish	Lemmatized, split	27.1	
2b. Swedish	Lemmatized, no split	17.4	-35.8
2c. Swedish	Stemmed	19.0	-29.9
3a. German	Lemmatized, split	31.0	
3b. German	Lemmatized, no split	26.4	-14.8
3c. German	Stemmed	25.7	-17.1

21.5 Discussion and conclusions

In this paper we focused on the question of the effectiveness of morphological processing in mono and cross-lingual IR. In our Monolingual IR tests we found out that, in Finnish IR, lemmatization by FINTWOL outperforms other approaches, in particular plain words and stemming, while inflectional stem generation approaches the performance of lemmatization. Their difference in performance is not significant. However, in the latter approach, the index must be harvested for full words matching the generated stems. Thus queries tend to become unmanageably long. Kettunen (2005) has however found that by extending the inflectional stems by regular expressions, query length can be reduced dramatically with only a minor penalty in performance.

In the second set of monolingual tests we found that the performance of lemmatization by FINTWOL when compounds were split vs. kept intact, splitting compounds clearly improved performance. Interestingly, in the test collection used, stemming by Snowball approached lemmatization in performance. In the English monolingual tests, stemming was found better than lemmatization by ENGTWOL. Simpler morphology and the lack of compound words in English compared to Finnish seem to explain the finding. However, another test collection might yield slightly different results.

In our cross-lingual IR tests, *English* was the source language, and *Finnish*, *German*, and *Swedish* served as target languages. In all findings, lemma-

tization and splitting compounds by TWOL clearly outperforms other approaches. This further confirms the importance of handling compound words properly in compound-rich languages. The relative performance of lemmatization without splitting compounds vs. stemming gave mixed results, which may in part be explained by the quality of stemmers.

In summary, lemmatization and splitting compounds in morphologically complex languages seem to consistently provide the best performance. The down sides are that this approach requires large dictionaries, which need to be updated, and techniques for handling the unavoidable and important out-of-vocabulary words. Automatic stem generation seems to be a much lighter-weight approach delivering competitive performance, at least in the case of Finnish. However, in this approach, after harvesting full index words matching the generated stems, queries tend to become long. This may be critical for efficiency in some IR environments. Further research in morphological processing for IR is therefore in order.

References

Abu-Salem, H., Al-Omari, M. and Evens, M.W. 1999. Stemming methodologies over individual query words for an Arabic information retrieval system. *Journal of the American Society for Information Science* 50(6): 524-529.

Airio, E., Keskustalo, H., Hedlund T. and Pirkola, A. 2003. Multilingual experiments of UTA at CLEF 2003: The impact of different merging strategies and word normalizing tools. Peters, C. and Borri, F. (eds.) *Results of the CLEF 2003 Evaluation Campaign, Cross-Language Evaluation Forum, Italy*, pp. 13 - 18.

Airio, E. 2005. Word normalization and decompounding in mono- and cross- lingual IR. *Information Retrieval*. To appear.

Alkula, R. 2000. *Merkkijonoista suomen kielen sanoiksi*. Acta Universitatis Tamperensis 763, Available at: http://acta.uta.fi/pdf/951-44-4886-3.pdf.

Alkula, R. 2001. From plain character strings to meaningful words: producing better full text databases for inflectional and compounding languages with morphological analysis software. *Information Retrieval* 4: 195 - 208.

Arppe, A. 1996. Information explosition and the use of linguistic tools in Finland. *Kieli ja Tietokone, AFinLAn Vuosikirja 1996*. Suomen Soveltavan Kielitieteen Yhdistyksen Julkaisuja, 54 (= AFinLA Series, 54), pp. 7-32.

Broglio, J., Callan, J., Croft, B. and Nachbar, D. 1995. Document retrieval and routing using the INQUERY system. *Proceedings of the Third Text Retrieval Conference (TREC-3), Gaithesburg*, MD: National Institute of Standards and Technology, special publication 500-225, pp. 29 - 38.

Callan, J., Croft, B. and Harding, S. 1992. The INQUERY retrieval system. *Proceedings of the Third International Conference on Databases and Expert Systems Applications*, Berlin: Springer Verlag, pp. 78 - 84.

Davis, M. and Ogden, W. 1997. QUILT: Implementing a large-scale cross- language text retrieval system. *Proceedings of the 20th Annual International ACM SIGIR*

Conference on Research and Development in Information Retrieval, Phidadelphia, PA, USA, pp. 92-98.

Frakes, W. 1992. Stemming algorithms. Frakes, W. and Baeza-Yates, R. (eds.), *Information Retrieval. Data Structures and Algorithms*, Prentice Hall, pp. 131 - 160.

Harman, D. 1991. How effective is suffixing? *Journal of the American Society for Information Science* 42(1): 7-15.

Hedlund, T., Keskustalo, H., Pirkola, A., Airio, E., and Järvelin, K. 2001. *UTA-CLIR @ CLEF 2001. Working Notes for CLEF 2001 Workshop*. Available at: http://www.ercim.org/publication/ws-proceedings/CLEF2/hedlund.pdf

Hull, D. 1996. Stemming algorithms: a case study for detailed evaluation. *Journal of the American Society for Information Science* 47(1): 70-84.

Kalamboukis, T.Z. 1995. Suffix stripping with modern Greek. *Program* 29(3): 313-321.

Karlsson, F. 1987. *A Finnish grammar*. Porvoo: WSOY.

Kettunen, K., Kunttu, T. and Järvelin, K. 2005. To stem or lemmatize a highly inflectional language in probabilistic IR environment. *Journal of Documentation*. To appear.

Kettunen, K. 1991a. Doing the stem generation with Stemma. J. Niemi (ed.), *Papers from the Eighteenth Finnish Conference of Linguistics*. Kielitieteellisiä tutkimuksia, Joensuun yliopisto, N:o 24, pp. 80 - 97.

Kettunen, K. 1991b. Stemma, a robust noun stem generator for Finnish. *Humanistiske Data* 1: 26 - 31.

Kettunen, K. 2005. Developing an automatic linguistic truncation operatorfor best-match retrieval in inflected word form text database indices. Submitted to Information Retrieval.

Koskenniemi, K. 1983. *Two-level morphology: a general computational model for word-form recognition and production*. Publications of the Department of General linguistics, University of Helsinki. No. 11.

Koskenniemi, K. 1985. FINSTEMS: a module for information retrieval. Karlsson, F. (ed.), *Computational morphosyntax. Report on research 1981 - 84*. Publications of the Department of General linguistics, University of Helsinki. No. 13., pp. 81 - 92.

Krovetz, R. and Croft, W.B. 1992. Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems* 10(2): 115-141.

Krovetz, R. 1993. Viewing morphology as an inference process. *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Pittsburg, PA, pp. 191-202.

Kunttu, T. 2003. *Perus- ja taivutusmuotohakemiston tuloksellisuus todennäköisyyksiin perustuvassa tiedonhakujärjestelmässä*. Informaatiotutkimuksen pro gradu - tutkielma. Informaatiotutkimuksen laitos, Tampereen yliopisto. (M.Sc. Thesis, University of Tampere, Dept. of Information Studies).

McNamee, P. and Mayfield, J. 2004. Character n-gram tokenization for European language text retrieval. *Information Retrieval* 7(1-2): 73-97.

Oard, D. and Diekema, A. 1998. Cross-language information retrieval. *Annual Review of Information Science and Technology (ARIST)* 33: 223-256.

Pirkola, A. 1998. The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. *Proceedings of the 21st Annual International ACM Sigir Conference on Research and Development in Information Retrieval*. Melbourne, Australia, pp. 55-63.

Pirkola, A., Hedlund, T., Keskustalo, H., and Järvelin, K. 2001. Dictionary- based cross-language information retrieval: problems, methods, and research findings. *Information Retrieval* 4(3/4): 209-230.

Pirkola, A. 2001. Morphological typology of languages for IR. *Journal of Documentation* 57(3): 330 - 348.

Popovic, M. and Willett. P. 1992. The effectiveness of stemming for natural- language access to Slovene textual data. *Journal of the American Society for Information Science* 43(5): 384-390.

Porter, M.F. 1980. An algorithm for suffix stripping. *Program* 14: 130-137.

Porter, M. 2001. *Snowball: A language for stemming algorithms*. Available at: http://snowball.tartarus.org/texts/introduction.html

Sanderson, M. 1994. Word sense disambiguation and information retrieval. *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland*, pp. 142-151.

Savoy, J. 1999. A stemming procedure and stopword list for general French corpora. *Journal of the American Society for Information Science* 50(10): 944-952.

Sormunen, E. 2000. *A method for measuring wide range performance of Boolean queries in full-text databases*. Acta Universitatis Tamperensis 748. Tampere: University of Tampere.

22

A Grammar for Finnish Discourse Patterns

KRISTIINA JOKINEN

22.1 Introduction

This article deals with Finnish discourse oriented word-order variations, and provides their implementation in the HPSG-style typed feature structure grammar using the LKB toolkit (Copestake, 2002). It does not present a full-coverage Finnish grammar or even a small HPSG fragment of the standard syntactic phenomena in Finnish. Rather, the aim has been to implement the Finnish discourse configuration in the Finnish Discourse Pattern Grammar (FDPG), employing typed feature structures and old and new discourse information, and thus to supply a starting point for further research in computational modelling of syntax-discourse interplay. The goal is motivated by the need for a dialogue system to analyse utterances and generate responses using semantic representation which is rich enough to encode discourse referents with different information status. The dialogue manager distinguishes old and new information, keeps track of the discourse topic, and also provides a context e.g. for the specific corrections where the speaker objects what has been stated in the previous utterance and contrasts it with a new fact. The use of topic and new information in language generation is discussed in Jokinen and Wilcock (2003) in more detail.

The interpretation of the Finnish word-order variations is based on Vilkuna (1989). She points out that the different syntactic orders reflect a discourse configurational structure of the language: constituents in certain positions are always interpreted as conveying particular discourse functions. In order to parse the word-order variations in the HPSG grammar formalism, I will argue

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

in favour of discourse *patterns*. These are fixed orders of the main sentential constituents based on Vilkuna's discourse configuration and used for presenting and interpreting discourse information in utterances. I have extended the head-complement and head-specifier rules in the HPSG grammar with a set of combination rules that concern discourse patterns, so that the patterns can be effectively used in parsing the various word orders.

The article is organized as follows. I will first review Vilkuna's discourse configuration for simple transitive sentences and discuss its relation to the information structure. This is followed by a short introduction to HPSG, the LKB formalism, and typed feature-structures. I will then present the implementation of the discourse patterns in LKB, and finally discuss some points for further research.

22.2 Finnish Discourse Syntax

22.2.1 Word-order variations

Vilkuna (1989) defines the following discourse configuration for Finnish:

| Kontrast | Topic | Verb | Rest |

The main verb divides the sentence into two parts. The positions in front of the verb carry special discourse functions while the Rest-field after the verb contains constituents in no particular order. (The end of the sentence, however, marks new information, see below.) The two specific discourse functions are Kontrast (K) and Topic (T), assigned to the elements occupying the sentence-initial position (K) and the position immediately in front of the main verb (T). The T-position marks the current discourse topic, i.e. what the sentence is about. The K-position can be occupied by a discourse referent which is contrasted with the topic of the previous sentence. It is always a marked position with contrastive emphasis, and it can also be empty.

In order to determine the information status of the constituents, the Prague school question-answering method is used: one seeks for a suitable question that the sentence provides new information for, and the information status of the constituents is determined in relation to this context. Notice that in dialogues, answers typically realize only the new information, since Topic and discourse-old information can be inferred from the previous utterance and discourse context (Jokinen and Wilcock, 2003). If the utterance has K-position filled, the underlying discourse context does not contain a question but rather a statement that is contrasted or corrected, see examples below and in Section 22.4.2.

For a simple transitive sentence, the following alternatives are possible:

	Kontrast	Topic	Verb	Rest	English equivalent
1		Karhu	pyydysti	kalan	The bear caught the fish
2		Kalan	pyydysti	karhu	The fish is caught by the bear
3	Kalan	karhu	pyydysti		It is the fish that the bear caught
4	Karhu	kalan	pyydysti		It is the bear that caught the fish
5	Pyydysti	karhu		kalan	The bear DID catch the fish
6	Pyydysti	kalan		karhu	

Sentence (1) represents the canonical word order for Finnish: it has subject in the T-position and object in the Rest-field. Statistically it is also the most common word order, supporting the fact that the subject usually encodes the topic. As for the information structure, three alternatives are possible: the whole event can be new as in the presentation sentence (“What happened?”), the verb phrase can be new (“What did the bear do?”), or only the object can be new (“What did the bear catch?”). The sentence (2) is analogous, except that the constituents have now swapped places: the object is Topic while the subject introduces new information in the discourse. The utterance matches the question “Who caught the fish?”

Sentences (3) and (4) signal correction in regard to the previous discourse. They pair up so that the sentence initial K-position is occupied by the object/subject which is contrasted with another object/subject mentioned earlier in the discourse: e.g. “It is the fish that the bear caught, not an otter”, and “It is the bear that caught the fish, not the wolf”[1]. The sentences (5) and (6) have a special argumentative character, too, since the main verb is in the K-position. In (5), the speaker insists on the truth of the statement ("indeed the bear did catch the fish"), but the word-order is also used if the speaker presents the state of affairs as new, something surprising and contrary to expectations (*no, pyydystin minä pienen kalan* “well I did catch a small fish”). The alternative (6), however, with the object occupying the T-position, is awkward in simple sentences. Obviously this is due to the clash of the two specially marked word order patterns: the preposed and contrasted verb does not fit with the marked word order that indicates the subject as new information.

22.2.2 Information structure

Discourse configuration bears similarity to information packaging (Engdahl and Vallduví, 1996), although it does not exactly correspond to the sentential information structure. As Vallduví and Vilkuna (1998) point out, contrastivess is orthogonal to information structure. While the elements in the Rest-field are new (rheme) and the elements in the T-position are old and carry presupposed information (theme), the information status of the K-position is not so clear; cf. also the failure of the question-answer method to directly provide a

[1]Kontrast can also be expressed by intonation in the neutral SVO order: *Karhu pyydysti KALAN*, or *Kalan pyydysti KARHU*. I will not discuss them further here.

context for the sentences (3)-(6) above: the context contains statements rather than queries for new information. Kontrast is of course new with respect to the sentential content, but it can also be old, if the referent has already been introduced in the discourse context. For instance, (4) can occur after the discourse like "I saw a wolf and a bear by the lake" - "and it was the wolf that caught a fish?" - "No, not at all, it was the bear that caught the fish, not the wolf". In fact, in this case we have a curious situation where a discourse referent is simultaneously old and new; Vilkuna (1989) calls these Topic-Focus cases. In FDPG, discourse referents in the K-position are regarded as new, since to the hearer, contrast is new information, and the discourse referent that turns the proposition into a new fact is the one occupying the K-position.

I have previously (Jokinen, 1994) introduced Topic and NewInfo as two mutually exclusive features to distinguish two types of discourse referents: Topic represents what the utterance is about and NewInfo what is new in the discourse context. NewInfo is related to Topic: it describes something new with respect to the discourse topic. If the whole event is new, the discourse referent for the verb is marked as NewInfo, and we have a presentation sentence. The distinction agrees with that proposed by Vallduví & Vilkuna (1998), who describe topic as an anchor to the focus (new information). I will not go into details of semantic representation of Topic and NewInfo, but refer to Wilcock (this volume) who discusses different representations for information structure and indicates how Minimal Recursion Semantics can be extended to take information structure into account.

22.3 LKB, HPSG, and FDPG

22.3.1 Preliminaries

The first implementation of the basic Finnish word-order variations is presented in Karttunen and Kay (1985). They describe a parser for free-word order languages, and use functional unification grammar marking topic and new information as specific features on the constituents. For FDPG, I have used LKB[2] as the development tool. This is an open source grammar toolkit for implementing natural language grammars in the typed feature structure formalism. Most implementations in LKB use HPSG, but the LKB itself is powerful enough to allow grammars in any constraint-based linguistic formalism to be developed. The grammar files include lexicon (lexical entry definitions), rules (feature structures describing how signs can be unified), and types (type specifications that constrain on sign unification). The toolkit consists of various tools for the developer to write and debug grammars, and it comes with several sample grammars as well as a full stepwise course for learning how to build grammars.

[2] http://www.delph-in.net/lkb/

HPSG (Head-Driven Phrase Structure Grammar, Pollard & Sag 1994) is a strict lexicalist approach to human language modelling. It assigns rich information structures to words, and using various constraints, it projects phrasal categories and sentences from the words. Lexical heads specify information like part-of-speech and dependency relations, and also encode the basic semantic information of their phrasal projections.

The representation of lexical items, like that of the projected phrases is a uniform feature structure called the *sign*. A sign consists of attributes and their values, encoding phonological, morpho-syntactic, semantic and pragmatic information of the entries. In the ERG grammar[3], a sign contains the following features: ORTH (orthographical realization of the lexical sign), SYNSEM (syntactic and semantic information), LEX (lexical status), NON-LOCAL (non-local information), and HEAD (head information). The representation of signs is made more compact by organizing them into an *inheritance hierarchy*, according to which the signs can inherit certain properties from the more general entries above them in the hierarchy. Inheritance hierarchies are based on *typed feature structures*: each sign is associated with a type which constrains free unification of the otherwise compatible signs.

In HPSG, the arguments of lexical entries are divided into complements and specifiers. The two main rules that can be applied to lexical entries to form phrasal projections are the *head-complement rule*, which unifies the sign of the lexical head with the signs of its complements, and the *head-specifier rule*, which forms a saturated phrase by unifying the phrasal sign with the sign of the specifier. The sentential specifier is the subject, and the syntactically saturated phrase is a sentence.

The complement list is encoded in the lexical signs COMPS-feature, and the specifier specification into the SPR-feature, both of which are SYNSEM-features. The complement list and the application of the head-complement and the head-specifier rules are ordered, so possible word order variations must be described by other means. A simple solution is to allow multiple verb entries, one for each different word order that the lexical item can project. However, this explodes the lexicon, and for languages like Finnish, it is not a reasonable alternative. One can also introduce a special permutation rule that produces necessary variations in the COMPS-list for any lexical entry. The problem with this alternative is that it increases grammar processing time. Yet another solution is to use unordered sets as COMPS-features instead of lists. For instance, the Japanese JACY grammar[4] orders possible argument structures into a type hierarchy and allows different head-complement rules to pick up the arguments in the COMPS-list in any order.

[3]The LinGO English Resource Grammar,see http://www.delph-in.net/erg/

[4]Homepage: http://www.dfki.de/ siegel/grammar-download/JACY-grammar.html

In Finnish, however, the driving force in sentence analysis and generation seems to be discourse configuration rather than syntactic constituency. FDPG thus uses discourse patterns, defined in terms of the discourse functions Kontrast, Topic, and NewInfo as the main elements in describing Finnish word-order. Moreover, in a well-formed utterance, NewInfo must always be explicitly present.

In their HPSG account for English and Catalan information packaging, Engdahl and Vallduví (1996) suggest that the specific feature INFO-STRUCT, comprising the features FOCUS and GROUND, with the latter being further divided into LINK and TAIL, be added in the context field of lexical signs so as to encode sentential information structure. Following this, I also assume that the discourse configuration is a separate dimension in sentence analysis. However, instead of the feature INFO-STRUCT, I use DISC-STRUCT with the features KONTRAST, TOPIC, and NEWINFO[5].

Furthermore, I have replaced the head-complement and head-specifier rules by a set of special rules that describe how various discourse patterns can be combined and interpreted as the speaker/hearer (incrementally) parses the particular word order variations. The patterns can be ordered in a type hierarchy, although this is not done in the sample grammar. Since the focus of the article is on word order only, I have also made some simplifying assumptions about the morphosyntactic properties of Finnish:

– Morphology is encoded in the lexical entries,
– NPs require determiner.

22.3.2 The Finnish grammar categories

FDPG regards *utterance* as the smallest unit in syntax, emphasizing its occurrence as part of the discourse and being uttered by a speaker. The utterance sign has two fields: DISC-STRUC for discourse configuration, and SYNSEM-STRUC for the syntactic arguments and their semantics. The sign is of type *utt-struc*, and defined as follows:

```
utterance := utt-struc &
[ DISC-STRUC disc-struc,
  SYNSEM-STRUC synsem-struc,
  ARGS *list* ].
```

The DISC-STRUC contains the following features (EVENT encodes the event reference denoted by the main verb):

```
disc-struc := utt-struc &
[ KONTR *list*,
  TOPIC *list*,
```

[5]NEWINFO relates to FOCUS, and TOPIC to LINK, but KONTRAST has no apparent counterpart.

```
  NEWINFO synsem-struc,
  EVENT evtype,
  RESTFIELD *list* ].
```

The SYNSEM-STRUC contains feature representations for the lexical head, its syntactic dependents and semantics, and the DEP-feature includes the typical HPSG specifier and complement-lists:

```
synsem-struc := utt-struc &
[ HEAD pos,
  DEP dependents,
  SEM semantics ].

dependents := feat-struc &
[ SPR *list*,
  COMPS *list*].
```

Finally, phrases are projections of lexical items such that all of their complements have been found, i.e. the COMPS-list is an empty list.

```
phrase := utterance &
[ SYNSEM-STRUC [ DEP [COMPS < > ] ] ].
```

FDPG also introduces a special `subj-phrase` which has an empty SPR-list. Although specifiers do not mark saturated phases as is the case in HPSG in general, their separate marking has been retained, however, since in some discourse patterns the subject can be combined with the main verb before the other complements, and thus it is necessary to distinguish signs which have been unified with their subject from those that still have to find it.

```
subj-phrase := utterance &
[ SYNSEM-STRUC [ DEP [SPR < > ] ] ].
```

Some lexical verbs do not usually take nominative-case subjects (emotional and physiological states, nature descriptions), and accordingly, their lexical entries have SPR-feature instantiated to null. Some sentence types (existential and possessive sentences) do not have nominative-case subjects either, and in this case, it is the main verb `olla` "be" that has SPR-feature as null.

22.4 Discourse Patterns

The discourse patterns are rules which define how the lexical signs can be unified with their arguments, and they can be seen as an extension of the HPSG head-complement and head-specifier rules. The FDCG idea is to unify the lexical verb sign first with the immediate left/right adjacent NP sign, regardless of whether the NP has subject- or object-marking. This unification

serves as the basic pattern for marking discourse functions. The rest of the discourse functions are filled in when the second argument is unified with the basic pattern, resulting in a saturated phrase.

22.4.1 Head-complement patterns

The head-complement patterns deal with the pair-wise combination of the main verb (lexical head) with the adjacent NP sign (be it an NP functioning as subject or object), and they correspond to the HPSG head-complement rules. For space restrictions, the full feature structure is given only for the first pattern. `Subj` stands for a nominative case NP and `Obj` for an accusative NP. All examples are based on the event "the bear caught the fish".

1) Subject-Topic pattern:

Subj + V *karhu pyydysti* "the bear caught"

The rule combines the main verb with a nominative case NP on the left. The result is a feature structure for an utterance with both the NP and the main verb marked as Topic: `Topic(Subj) + Topic(V)`

```
subj-top-rule := utterance &
[ DISC-STRUC [ TOPIC < #1, #2 > ],
  SYNSEM-STRUC [ORTH #orth,
                HEAD #head,
                DEP [ SPR < >,
                      COMPS #comps ],
                SEM #sem ],

  ARGS < phrase & #1 & [SYNSEM-STRUC
                        [HEAD noun & [ AGR [ CASE nom ]],
                         DEP [SPR < > ] ] ],
         word & #2 &
         [ORTH #orth,
          HEAD #head & verb,
          DEP [ SPR < #1 >,
                COMPS #comps ],
          SEM #sem ] > ].
```

2) Object-Topic pattern:

Obj + V *kalan pyydysti* "the fish was caught by"

The rule is parallel to (1), but it combines the main verb with an accusative case NP (object). Also in this case, the resulting utterance has both the NP and the main verb marked as Topic: `Topic(Obj) + Topic(V)`

3) Verb-Kontrast pattern:
V + Subj *pyydysti karhu* "caught the bear"

This rule combines the verb with a nominative case NP which is now on the right. The result is an utterance where the main verb is marked as Kontrast and the subject-NP as Topic, i.e. something the utterance (and the correction) is about: `Kontrast(V) + Topic(Subj)`. The pattern anticipates the speaker's disagreement and wish to express a correction of what has been stated before.

4) Verb-New pattern:
V + Obj *pyydysti kalan* "caught the fish"

The rule is parallel to (3) and combines the verb with an accusative case NP on the right. However, the result has both the verb and the object-NP as new information: `NewInfo(V) + NewInfo(Obj)`. Notice that the verb is not interpreted as Kontrast like in (3), since this would lead to awkward interpretations when the subject-NP is unified with the phrase (see discussion about the sentence (6) in Section 22.2.1, and the impossible combinations in Section 22.4.3).

22.4.2 Head-specifier patterns

The head-specifier patterns are analogous to the HPSG specifier rules which saturate the phrasal sign with the specifier, i.e. in the case of a verbal sign with the subject-NP. However, in FDPG, the rules do not apply only to subject-NPs but also to object-NPs, and also the dicourse context comes to play a role in the unification. A rule can be applied only if the information status of the NP accords with that of the underlying discourse pattern. Thus the head-specifier patterns guide the parsing and constrain acceptability of a particular word-order with respect to the appropriate information structure of the utterance. Suitable discourse contexts are shown after each rule by an underlying question or statement that the resulting utterance addresses to. The abbreviations `SV, OV, VS,` and `VO` refer to the constituents formed by the patterns (1-4) above, and the utterances to the example utterances in Section 22.2.1.

1a) [Subject-Topic] Object-New pattern (utterance 1):
SV + Obj *karhu pyydysti + kalan* "the bear caught + the fish"

The rule produces one of the three information structures for the canonical word-order as discussed in Section 22.2.1 (the other two word-orders are produced by Rule 4a below). The basic SV pattern is already determined as Topic

(Rule 1), so the resulting utterance is used to introduce a new object-NP. The utterance thus functions as a response to an underlying question “What did the bear caught?” with the object-NP as NewInfo “(the bear caught) the fish”: *Mitä karhu pyydysti? – (karhu pyydysti) kalan*

1b) [Subject-Topic] Object-Kontrast pattern (utterance 3):
Obj + SV *kalan + karhu pyydysti* “the fish + the bear caught”

The unification of the basic SV pattern with the object-NP on the left results in an utterance where the object-NP is Kontrast. The SV pattern is determined as Topic as above(Rule 1), but the previous context has also presented an object which the speaker wants to contrast and correct. For instance, the previous context may contain a statement like “the bear caught the otter”, and the speaker then corrects this with a new object: *Eipäs, kalan (karhu pyydysti)* “no, it was the fish (that the bear caught)”.

2a) [Object-Topic] Subject-New pattern (utterance 2):
OV + Subj *kalan pyydysti+karhu* “the fish was caught by +the bear”

The rule is parallel to (1a) above: now the basic OV pattern is Topic (Rule 2), and the subject-NP on the right is introduced as new. The underlying question is “Who caught the fish?” with the subject marked as NewInfo "(The fish was caught by) a bear": *Kuka pyydysti kalan? - (kalan pyydysti) karhu*.

2b) [Object-Topic] Subject-Kontrast pattern (utterance 4):
Subj + OV *karhu+kalan pyydysti* “the bear + the fish was caught by”

Analogously to (1b) above, unification of the topical OV pattern with a subject-NP on the left results in a correction and a contrastive utterance. A previous statement, like “the wolf caught the fish”, is contrasted with the new subject-NP as Kontrast: *Eipäs, karhu (kalan pyydysti)* “no, it was the bear that caught the fish”.

3a) [Verb-Kontrast] Object-Rest pattern (utterance 5):
VS + Obj *pyydysti karhu + kalan* “caught the bear + the fish”

The rule combines the VS pattern with an object-NP on the right. The verb is already marked as Kontrast and the subject as Topic (Rule 3), and the object falls in the Rest-field.

The object can be either discourse old or new information. If the object-NP is old, the contrast concerns the actual event which the subject and the ob-

ject denote the participants of. The utterance occurs in contexts where doubts about the truth of the speaker's assertion have been raised ("I wonder if the bear caught the fish after all"), and the speaker wants to reinforce and insist on the original argument: "yes indeed the bear DID catch the fish": *Enpä usko että karhu pyydysti kalan. – Kyllä toki pyydysti karhu kalan.*

The object-NP can also be NewInfo as in the answers to nosy questions: *ja pyydystitkös mitään? – no, pyydystin minä pienen kalan* "and did you catch anything? - well, I did catch a small fish". This is a rather common pattern, since the speaker provides new information as a response to a genuine question. The contrast in this case concerns the implicit negative presupposition of the question "you may not have caught anything", which is contrasted by the speaker's positive answer. It should be noticed that the question "What did you catch?" presupposes that the partner caught something, while the question "Did you catch anything?" lacks such a presupposition. It is interesting that Finnish reflects the difference in the presuppositions in the word-orders that the possible answers to the these questions exhibit: the former is encoded in the Object-New rule (1a) and the latter in Object-Rest rule (3a).

4a) [Verb-New] Verb-Presentation pattern (utterance 1):
Subj + VO *karhu + pyydysti kalan* "the bear+caught the fish"

The rule produces two of the three canonical word-orders (cf. Rule 1a). The new information in the verb-object pattern (determined by Rule 4) is combined with the two possible discourse statuses of the subject-NP. If the subject-NP is Topic, the resulting utterance is simply an answer to the question "what did S do?": *Mitä karhu teki? – (Karhu) pyydysti kalan.*

If the subject-NP is new in the discourse context, the result is a presentation sentence with all the constituents as NewInfo, answering to the question "what happened?": *Mitä tapahtui? – Karhu pyydysti kalan.*

22.4.3 Impossible combinations

3b) Impossible-Verb-Kontrast-Object-Kontrast combination
Obj + VS

A symmetrical rule for (3a) would combine a contrasted verb and a topical subject-NP with an object-NP on the left. This is impossible, however, since the K-position is always sentence-initial, and there is "no space" left for a second Kontrast in front of the already contrasted verb. It must be noticed that the Obj-V-Subj word-order is fine if the discourse configuration is different: see the Object-Topic-Subject-New rule above (2a).

4b) Impossible-Verb-Two-News combination
NewInfo(VO) + NewInfo(Subj)

The other impossible combinations are based on the VO-pattern with the subject-NP on the right: `VO + Subj`.

If the VO-pattern is discourse new (introduced by Rule 4), it would seem natural to add subject-NP as new information at the end of the utterance. However, the combination is confusing: the resulting word-order is the marked discourse pattern for a sentence-initial Kontrast, while the all-new information status of the constituents suggests a presentation sentence. There is no Kontrast, and for a presentation sentence, the canonical SVO-order is preferred (rule 4a); hence unification is impossible. Even if we assume that the verb indeed is Kontrast and object-NP is NewInfo, Kontrast(V) + NewInfo(O) + NewInfo(Subj), the combination would still lead to confusion, since there is no Topic to anchor the contrast to.

4c) Impossible-Verb-Kontrast-Object-Topic-New-Subject combination
Kontrast(V) + Topic(O) + NewInfo(Subj)

If the verb is Kontrast and object-NP is Topic, the combination is analogous to the contrasting VS + Obj pattern licensed by the rule (3a). However, the order seems to favour the reading of the subject as NewInfo, and thus its interpretation is again confusing between whether the utterance is about contrasting events or a new subject. In the former case, the preferred combination would be VS+O (Rule 3a) and in the latter case OV+S (Rule 2a).

4d) Verb-Object-Topic-New-Subject combination
Topic(VO) + NewInfo(Subj)

If the VO-pattern is Topic, the subject must be NewInfo. In simple sentences, this combination will again run against the marked sentence-initial Kontrast pattern as well as the preferred Subject-New rule for introducing new subjects (Rule 2a). However, if an adverbial is added in the beginning of the utterance, the order becomes acceptable, although there are strong expectations that the contrast now continues with respect to the adverbial: *eilen pyydysti kalan karhu, tänään koira* "yesterday it was the bear that caught the fish, today the dog". The pattern suggests that there are two pairwise Kontrasts: the adverbials on the one hand and the subject-NPs on the other hand. The verb and the object-NP make up the topical background for the Kontrasts. In the current FDPG, which only deals with simple sentences, these combinations are not possible. Obviously, the grammar rules should be relaxed in order to allow

the unification of the constituents to take place, but they should also constrain the result to be unsaturated, so as to force the adverbial Kontrasts to be added in the sign and make the phrase well-formed from the discourse point of view. These combinations will require more detailed research.

22.5 Discussion

In this article I have presented a HPSG-based implementation of the Finnish word-order variations in the LKB grammar environment. Following Vilkuna (1989), the Finnish syntax is characterized by its discourse configuration, which assigns certain discourse functions (Kontrast, Topic, NewInfo) to the constituents according to their position with respect to the main verb. The Finnish Discourse Pattern Grammar (FDPG) can parse simple intransitive and transitive sentences and produce appropriate discourse interpretations of the different word-orders without spurious parses. The grammar is available on request from the author.

The combination rules are based on simple discourse patterns concerning the main verb and its adjacent NP-complements. There are different patterns for combining the main syntactic constituents: four for pair-wise combinations of the main verb and an adjacent NP, and six for producing saturated utterances with the discourse functions appropriately filled in. The former extend the head-complement rule of the traditional HPSG, while the latter extend the HPSG head-specifier rule. The patterns with their associated discourse functions can also be thought of providing guidance for the hearer about what to expect next in the on-coming discourse.

In Finnish, phrase structure thus seems like an epipheral phenomenon that occurs as a side effect of the lexical entries being projected into full sentences and their dependents ordered into a coherent discourse. From this view-point, it would, of course, be more natural to describe syntactic relations with the help of a dependency grammar which explicitly reveals the dependency relations between the verb and its arguments, than with a phrase structure grammar which focusses on phrasal structures. In fact, the patterns can be also seen as possible ways to combine dependency relations into surface strings.

I have effectively proposed a new approach to syntax: that of discourse configuration. In this approach the speaker's intention to exchange new information on a particular topic is taken as the driving force for communication, and this intention is not realized on the level of dialogue organisation only, but trickles down to the syntactic structure as well. Discourse information, carried by the different word orders in Finnish, is thus efficiently used by the hearers when processing the incoming utterance: the presence of certain discourse patterns directs the hearer to expectations concerning upcoming elements. Cognitive studies also seem to support psychological reality of discourse pat-

terns, their incremental processing and impact on the hearer's expectations about the yet-to-come elements. For instance, in a recent study of processing Finnish word-order variations, Kaiser and Trueswell (2004) found empirical evidence that shows how the hearers make efficient use of the non-canonical word order patterns to predict upcoming referents and their discourse status.

Finally, the FDPG is an attempt to provide a model for discourse configurational syntax that would describe the link between the syntactic-semantic representation of utterances and the information they encode of dialogue situations. The view of dialogue events as the determining factor for sentence structuring may prove useful in modelling also other "free" word-order languages like Japanese, where the discourse function Topic is grammaticalised and none of the verb arguments are obligatory in well-formed sentences. Of course, systematic investigations are needed to substantiate this hypothesis.

22.6 Acknowledgments

The research was conducted while I was a Visiting Fellow in Clare Hall at the University of Cambridge, associated with the NLP group in the Computer Laboratory. I would like to thank Ann Copestake for her assistance and useful discussions concerning LKB implementation, Karen Spärck Jones for providing an encouraging environment, and Graham Wilcock for discussions on HPSG grammars in general.

References

Copestake, A. 2002. *Implementing Typed Feature Structure Grammars*. Stanford: CSLI Publictions.

Engdahl, E. and Vallduví, E. 1996. Information packaging in HPSG. In C. Grover and E. Vallduví, eds., *Edinburgh Working Papers in Cognitive Science, Vol. 12: Studies in HPSG*, pages 1–32. University of Edinburgh.

Jokinen, K. and Wilcock, G. 2003. Adaptivity and response generation in a spoken dialogue system. In J. van Kuppevelt and R. Smith, eds., *Current and New Directions in Discourse and Dialogue (Text, Speech and Language Technology, Vol. 22)*, pages 213–234. Kluwer Academic Publishers.

Karttunen, L. and Kay, M. 1985. Parsing a Free Word Order Language. *Natural Language Parsing*, eds. D. Dowty, L. Karttunen and A. Zwicky, 279-306. Cambridge: Cambridge University Press.

Kaiser, E. and Trueswell, J. C. 2004. The role of discourse context in the processing of a flexible word-order language. *Cognition 94(2)*, 113-147.

Vallduví, E. and Vilkuna, M. 1998. *On Rheme and Kontrast. The Limits of Syntax*, eds. P. Culicover and L. McNally, 79-109. New York: Academic Press.

Vilkuna, M. 1989. *Free Word Order in Finnish. Its Syntax and Discourse Functions*. Helsinki: Suomalaisen Kirjallisuuden Seura.

Wilcock, G. 2005. Information Structure and Minimal Recursion Semantics. (this volume)

23

Meaningful Models for Information Access Systems

JUSSI KARLGREN

23.1 Distributional models of language

Study of semantics has the general goal of modeling human linguistic competence as a theory, probing the constraints and limitations of language as a system of expression and representation, and of providing language engineering applications with a model of meaning, appropriate to its tasks. In general, there is no need to design a semantic model intended for practical processing to be neurologically or psychologically plausible but since human performance is impressive in certain respects there certainly is reason to investigate it to find if it can provide inspiration, examples, or constraints for implementations. Human information processing is efficient and effortless. The human information processor is flexible, dynamic, ever learning, does not stumble at inconsistencies, and does not require formal or explicit instruction.

What sort of demands would we want to pose on a model of meaning, from the standpoint of language engineering for information access? Some specific requirements are at the forefront for information access analysis. Information access involves matching brief or even incomplete expressions of information need to relatively more verbose documents and items of information. The documents are not necessarily formulated for ease of retrieval in mind.

For this class of tasks, models that are based on dynamically observed data of language use in some form are dominant. They have common characteristics, however those data are collected and whatever the character of the data: they are based on occurences of linguistic units in a context of use; they do not rely on explicitly represented pre-compiled knowledge; they are flexible

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

and sensitive to the domain and universe of discourse at hand.

The *Distributional Hypothesis*, the basis for distributional language models, states that two words are similar to the extent that they share contexts Harris (1968), and thus that distributional data — of how words appear in contexts — can be used to model similarity, however it is understood, between words. That statement can be used as a basis for a theory of meaning suitable for practical deployment in contexts where approximative semantic analysis of large amounts of linguistic data is necessary, approximating similarity in use with similarity in meaning.

Change or semantic drift is modelled seamlessly by distributional models. New data will provide new occurrence data for the model. The problem of modeling change can be formulated as the problem of selecting the right training context: what data are relevant to the model at hand? If the correct situational context is provided for the model, the resulting representation will reflect the usage in them. This is a desirable quality in the models: we know human language changes fluidly. From one intellectual context to another and from one discourse situation to another the usage and prototypical referents of expressions shift and change with little or no confusion for human users; as time passes, words' meanings evolve and change with little or no confusion, without any attention from their users.

Most distributional models are difficult to provide with precomputed data — to "teach" — in a non-arbitrary manner. Again, this is a desirable quality. We know people learn language their entire life. They do this *without explicit acts of definition and instruction*. In keeping with this it would be useful to find that a system for processing large amounts of text from varying sources have a semantic model capable of operation with little human intervention, with the necessary knowledge extracted from the data at hand. Distributional models in practice are implemented not only to work without supervision but in fact most often to forswear it entirely.

Most distributional models do not rely on external fixed knowledge sources to any great extent, and base their deliberations on statistical or probabilistic calculation on the data alone. We know people seldom take recourse in definitions or formal delimitations of meaning between types of expression. Expressions can be more or less similar in meaning, changing with author and reader perspective or situational context: a semantic model for robust processing of information from many authors to many readers must not be brittle and dependent on exact expression of formal knowledge — it should seamlessly incorporate the gradual shift in meaning from same to similar and from related to distinct (Karlgren, 1976, e.g.). Distributional models are typically implemented with calculation frameworks with intrinsic provision of gradual shades of *homeosemy* or relative similarity.

As can be inferred from the sketchy description above, both *word* or *term*

on the one hand and *context* on the other are central for modeling distributional data. The data may be preprocessed to identify graphical word occurrences, morphologically normalized words, multi-word terms, or whatever linguistic unit is being considered. The nature of the context studied varies according to what sort of model is being built: an utterance, a window of a few surrounding word tokens, an entire text, or a topical unit.

23.2 Representing distributional data — understanding language models

Distributional models collect data of term occurrences. These data are compiled in some representation for convenient further processing. *Probabilistic language models*, e.g., refine the occurrence data into an estimate of the probability that a given word will appear again, given some observed or observable context.

The dominant language model for analysis of textual information in information access and lexicographical applications is the *vector space model*. A vector space is a many-dimensional space where the points can be accessed by address – by a vector of coordinates using some system, typically cartesian. A point in a vector space can be described by a vector $\vec{v}$ thus:

$$\vec{v} = [v_1, \cdots, v_n]$$

where n is the dimensionality of the vector space.

The vector space model for languages posits such a many-dimensional space for terms by populating a vector space with distributional data of term usage in text or discourse. The data are represented in a matrix F of order $w \times n$, such that the rows F_w represent the terms, the columns F_n represent the contexts under consideration — documents, e.g., in the most typical case — and the cells are the (possibly weighted and normalized) frequency of a given term in a given context. Each row of frequency counts thus constitutes an n-dimensional occurrence vector $\vec{v}$ for a given term. These occurrence vectors, interpreted as coordinates in an n-dimensional space as above, deliver a vector space model with the occurrence vector defining a location for its term.

Vector space models have gained increasing currency for application to information access tasks. They exhibit several attractive qualities, not the least being that of pleasing intuitive simplicity, transparency and ease of explication. They are also computationally efficient in several respects, and have proven useful in several applications.

This model lends itself naturally to the application of standard distance metrics. Position is determined by the occurrence of terms in contexts; closeness in space implies distributional similarity or similar usage; and proximity between points — terms — in this space can easily be understood as simi-

$$d_{cos}(\vec{u},\vec{v}) = \frac{\vec{v}\cdot\vec{u}}{\mid \vec{u} \mid\mid \vec{v} \mid} = \frac{\sum_{i=1}^{n} v_i u_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}}$$

FIGURE 1 Computation of cosine between two vectors

larity in meaning. This notion of proximity or distance can be used to model gradual shades of relative similarity.

Similarity can be established either by calculating the distance between the points in space, or by transforming the vectors to polar coordinates and using the angle between them. This, in essence, normalizes the relative magnitude of the cell values in the matrix – vectors with the same orientation are considered equal. Most often the cosine of the angle as per the formula in Figure 1 is used: it interprets readily as a proximity measure.

In summary, vector space models localize terms at points in space. Proximity of a term to other terms is calculated through some distance measure. The meaning of a term is found by inspection of its closest neighbors — meaning is considered to be located in a region around terms. Terms can shift meaning, and this is modeled by moving the term to another point in space.

23.3 Space and meaning

As any model, the vector space model is intended to simplify the notion it is modeling, better to aid processing or understanding the object notion; as any metaphor the space and distance metaphor for meaning mediates experience from one area of human activity to another by conceptual transference.

The space metaphor is powerful and pervasive in human thinking and seems to fit in neatly with intuitions about how meaning comes about. Expressions such as “close in meaning” abound. But what sort of space do people think about when they use spatial expressions to discuss meaning?

While relative distance or proximity seem to be central, neither absolute distance measures nor other spatial relations are normally used. Each semantic comparison we make can be made in terms of proximity — no other relations are simple to make explicit. “Close in meaning.” or “Closer in meaning.” are acceptable statements; “*Slightly above in meaning.”, “*More to the north in meaning.” and “*One metre removed in meaning.” are not. It seems that our conception of meaning as space is limited to something like a limited view of a one-dimensional space.

23.4 Distributional models do not preserve all distributional information

While the distributional models base themselves on occurrences in data, they generalize from those observations, thus ridding themselves of overly specific

information. Probabilistic models sample the data and establish estimates of probable reoccurrence of observed items; vector space models compile the occurrence data into a point in vector space. In both cases, a large amount of distributional information is discarded.

The vector space model is useful and attractive, but does have limitations. Some of them have to do with our understanding of the space metaphor itself: the notion of distance between points leads us to the wrong calculations and an incorrect view of what the space is. While the multi-dimensional space *may* be the correct framework to solve structural problems of the representation, our intuitions risk leading us astray.

The intuitive use of the expressions "conceptual distance" or "close in meaning" does not specify in what way that distance is calculated, nor what topological status the locus of "concept" or "meaning" have; neither does the vector space model require a specific distance measure or definition of meaning. Yet the influence of our intuitions from living in two dimensions of a three-dimensional world via grade school geometry to the vector space calculations have led us to a too constrained view of what can be achieved using the model. This constraint may be inherent in the model, but it may also be a constraint only of the metaphor and our representation of the model. Determining whether the metaphor or the model is the limiting factor is difficult or impossible to do without proper calculation; our intuitions about space and meaning are not the right tools to make informed decisions.

The solar system metaphor of an atom is a parallel case of a representation and a model leading its users to wrong conclusions. The solar system model is seductive in its simplicity and its imaginative qualities. A considerable amount of effort in higher physics classes is spent trying to unlearn the model — which has been useful for gaining the first glimpses and first steps of understanding of subatomic structure, but where each obvious successive generalization is a step in the wrong direction.

23.5 Points, distances, and dimensions

Vector space models localize terms at points in space. Terms can shift meaning, which is evidenced by their occurrence data; these data are accommodated in the model by moving the term to another point in space. Relations to other terms change accordingly, and are evidenced by new distances calculated between them. This simple operation adheres well to our intuitions of how points in space can be manipulated. When modeling some types of observable distinctions in meaning made in human discourse it may well be contested in view of its discarding a considerable amount of information.

The study of vagueness, polysemy, generality, and other types of distributionally evident data would be well accommodated by broadening the scope

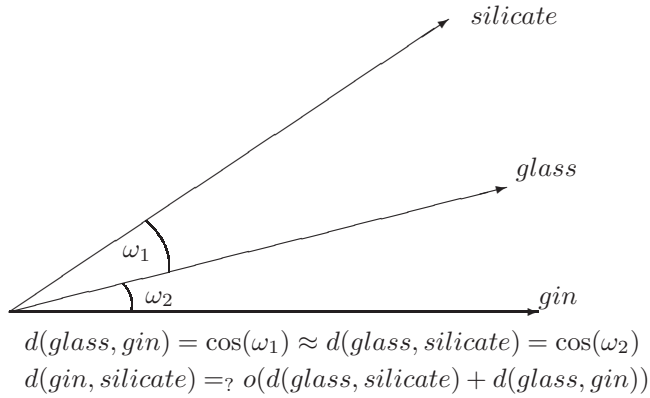


FIGURE 2 Polysemous terms have many kinds of neighbors in two dimensions

of how terms are represented in the model and attendant reform of how the notion of semantic distance is represented.

Distance between two points in a euclidean space is symmetrical and transitively calculable. This does not necessarily always have to be the case in a semantic space. Distance can be calculated in numerous ways. It is possible to examine the implementation of the space metaphor closely, and retool that implementation better to transcend our first intuitions of what geometry is to e.g. allow for non-euclidean, non-symmetric, non-transitive distance measures.

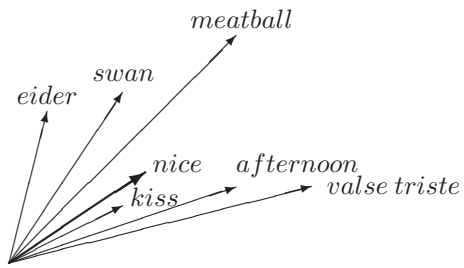
Polysemous terms are a case in point. Proximity between “glass”, the beverage, and “gin” on the one hand and between “glass”, the substance, and “silicate” on the other need not imply proximity betwen “gin” and “silicate”, as illustrated in Figure 2. The risk of confusing transitive proximities can be addressed within the standard term-as-points-framework using additional calculation — by retaining more distributional data in the model and allowing the term to occupy a trace or a more complex structure than a point in vector space.

Vague terms are another example. The capability of vector space models to handle the distinction between vague and definite usage is very limited. If a term in the data is used vaguely, the resulting representation will still try to pull the data together into a point. The representation of a term in the model does not in any way carry the information whether the term should be understood as definite or vague; the distance between terms is calculated identically from a point in vector space whether they are vague or specific. The model pulls together various items as exemplified in Figure 3. It can be argued that the model simply reflects the data: lots of things are nice, and they share a feature. The potential problem with the model is that the vague quality

of niceness is typically modeled as strongly as is the definite quality of, say, animacy or birdness.

In general, measurement of distance can in the given family of vector space models only be calculated between terms — which is of little utility given that the stated objective of most distributional models is to understand the relationship between concepts or whatever notional units of meaning one postulates. A term without a well-defined meaning — arguably the majority of terms — cannot be represented in any other way than as an (typically weighted) average of its occurrences. This distinction, if addressed at all, should be handled on model level. The vector space model does not handle this distinction.

It is not inherently necessary for the model to attempt to fold together the representation of each term into a point. It is a relatively simple extension to investigate terms represented by spaces rather than points, such as clouds, hyperplanes, clusters or concentric structures — it would involve simply imply retaining more data when refining the raw occurrence data and representing the additional data in the vector space. Higher-order distributional characteristics can be utilized to determine which geometry the distribution of a term should be modeled by: patterns of distribution can be modeled by patterns in space rather than using averages, which throw out most of the distributional information. Such an extension, however, will by necessity break the standard metaphor and its distance measure: the distance between two clouds is not well-defined from without the model itself, and needs to be addressed explicitly, not by inheritance via a metaphor.



$$d(kiss, swan) = o(d(kiss, nice) + d(swan, nice))$$

FIGURE 3 A vague term will be close to concrete terms

23.6 More meaningful models?

In conclusion, distributional models in general, and vector space models specifically, risk having their usefulness overshadowed by overly simple metaphors of use which constrain the amount of information extracted from the raw occurrence data upon which they are built. To better accommodate some of the features of the model or to investigate extended calculation bases of the model, higher-order data could be included — e.g. in some of the directions indicated above. By ridding the vector space model from the simple distance metaphor it is delivered with it will lose one of its most appealing qualities – that of pandering to our intuitions – but promises to gain in explicatory power.

23.7 Acknowledgments

The argument above has as its starting points discussions with Magnus Sahlgren, Pentti Kanerva, and Henrik Hållsten. Several valuable points on meaning and its representation are elaborately addressed by Dominic Widdows in a recent and lucid exposé (2005).

References

Harris, Zellig. 1968. *Mathematical Structures of Language*. Interscience publishers.

Karlgren, Hans. 1976. Homeosemy – on the linguistics of information retrieval. In D. E. Walker, H. Karlgren, and M. Kay, eds., *Natural Language in Information Retrieval - Perspectives and Directions for Research*. Stockholm: Skriptor.

Widdows, Dominic. 2005. *Geometry and Meaning*. California: CSLI Publications.

24

Word Senses

KRISTER LINDÉN

How many angels can dance on the point of a very fine needle,
without jostling one another?

— Isaac D'Israeli (1766-1848)

What is the meaning of a word? Unless one believes that we are born with an innate set of meanings waiting to find their corresponding expression in language, another option is that we learn the meaning of a word by observing how it is used by the language community we are born in. Some usages find their way into dictionaries and become established word senses. In order to understand what constitutes a word sense, we can look at the criteria lexicographers use when they decide that a word usage is a word sense and record it in a dictionary for future generations.

24.1 Language Philosophy

From a machine learning point of view Wittgenstein's suggestion (Wittgenstein, 1953) that "*the meaning of a word is its use in the language*" sounds plausible, because there is nothing else for a machine to observe. This view of meaning was made more specific by Harris, when he proposed that words with similar syntactic usage have similar meaning (Harris, 1954, 1968).

Even if we accept that the *potential* usage of words is unlimited, we are mainly interested in *real* usage when we learn to identify similarities or differences of word meaning. The real usage is prone to fluctuations and idiosyncracies, viz. usage preferences, of different language communities. A language community is any group of individuals who communicate. Some usage preferences become recognized by most communities of a language, a process known as lexicalization. Lexicalization progresses differently in dif-

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

ferent communities of a language giving rise to, e.g., synonyms.

The usage preferences as they manifest themselves in real usages characterize similarity or difference of word meaning. If someone says "Shoot!" when a bear is attacking, it is emotionally quite different from the same command when a small bird is flying by, although both require some weaponry. However, a reporter can shoot a question without extra equipment. For most usages of a written word, we do not have access to the full context, so there may be essential differences in other aspects than those in the text presented to a computer. Indirectly, by observing other usages of words in the context, it may still be possible for a computer to group the usages of *shoot* in 'shoot a bear', 'shoot a bird', and 'shoot a question' into two main groups of shooting with and without weapons. Then we present the machine with 'shoot a bullet' and expect the *bullet* to be more like a *question* than a *bear*, because in fact the main division does not really depend on the presumed weapon, but whether the direct object of *shoot* is animate or inanimate. We call this distinction a semantic feature. A multiple-inheritance taxonomy of such features is a feature structure. The animate and inanimate distinction is not fixed for every word, but may lend itself to modification or underspecification as in 'shooting stars'. A machine making observations based on a limited amount of samples of the real usage of a word in written text will end up with a piecewise approximation of features such as animate and inanimate.

24.2 Enumeration vs. Generation

The simplest way to create a dictionary of word senses is to enumerate each sense separately. If no further information is provided about how the senses are related, this representation requires each new sense to be manually added. A more flexible representation is presented by Pustejovsky (1998), a generative lexicon (GL), where the word senses are generated through the unification of feature structures guided by an inheritance system for the argument, event and qualia structures.

The GL is sometimes seen as a fundamentally different approach from the idea of dictionaries or lexicons as a simple enumeration of word senses, because the theory on generative lexicons claims that the GL also accounts for novel uses of words. Kilgarriff (2001) tested this claim on a set of corpus words and found that most of the novel or non-standard usages were unlikely to be accounted for by any GL, i.e., those usages that were not accounted for in a regular dictionary. The main benefit of a large-scale dictionary based on the GL theory would be that similar distinctions would consistently be made throughout the dictionary for all words with similar or related usages.

From a computer programming point of view, it is not particularly surprising that a lexicon program, i.e., a GL, is more flexible than a list of word

descriptions, more consistent and more compact, but equally unimaginative. In addition, as the GL grows, it is likely to be more unpredictable and more difficult to maintain. A GL comes with all the benefits and drawbacks of a large computer program and as such it covers only the words and senses it has been either intentionally or unintentionally programmed to cover.

24.3 The Origin of Features

A more fundamental problem related to language learning and child language acquisition is how we learn to associate meaning with sound sequences or words. We do not get closer to a solution for this problem by dividing a word into semantic features, because then we have to ask where the features come from or how they become primitives of the lexicon.

Interesting research on how meaning is associated with sound sequences has been done by Kaplan (2001) in his simulation of a robot society communicating about positions of several colored figures, i.e., circles, triangles and squares, on a white board using a Wittgensteinian language game. He was able to demonstrate that, when several stable language communities had evolved, synonymy arose. When the communities were in sporadic interaction, the communities kept their own words for the concepts but were able to understand other variants. By inspecting the robots he could determine that they had words for colors, shapes and relative positions. The robot simulations indicate that with suitable and not too complicated models, language can be learned from scratch in a language community interacting with the external world.

Research by (one of Harris' students) Gleitman (1990, 2002) on child language acquisition indicate that children learn nouns with external references before they learn verbs and then start distinguishing between different argument structures of the verbs. Her research supports the assumption that the meaning of verbs is tightly tied to their argument structure. The child language research gives some psychological relevance to the GL approach indicating that a GL is not merely a way of compressing the lexicon description.

If we accept that features and the meaning of features can be induced through language usage in a language community, a full-scale GL for some application would be an interesting effort both as a collection of linguistic knowledge and as a benchmark for future automatically induced vocabularies. It is quite likely that for some time to come high-performing computational lexicons will be partly hand-made with a generative component and a trainable preference mechanism[1]. A well-designed linguistically motivated

[1]On a parallel note, we quote Kohonen's personal comment on his self-organizing maps: "Once it has been shown that a map always organizes regardless of how random the initial state is, there is no need to show this every time. It is quite acceptable to speed things up by starting

GL with a trainable preference learning mechanism might be a good candidate for how to organize a word sense lexicon. There is no need for a computer to always learn the lexicon from scratch, despite the fact that this seems to be the way nature does it.

24.4 Recording Word Senses

New words and concepts arise at a steady pace and old words become associated with new meanings, especially in technology and biotechnology which are currently the focus of intense research efforts. In these areas specialized efforts like named entity recognition aim at identifying the meaning of new terms in the form of abbreviations, nouns and compound nouns by looking at their context. These entities are typically classified into names, dates, places, organizations, etc. Named entities and word senses represent two different aspects of the same problem. Named entities are usually new, previously unseen items that acquire their first word sense, whereas word sense discovery and disambiguation typically have assumed that words have at least two meanings or word senses in order to be interesting. It is, however, likely that the mechanism or process that attaches the first word sense to a string is the same as the one that later attaches additional meanings or word senses to the same string either by coincidence, i.e., homonymy, or by modifying some existing meaning, i.e., polysemy.

Other work on this theme distinguishes different word senses when a word gets different translations (Resnik and Yarowsky, 2000) so that the sense identification problem merges with finding appropriate translations. This analogy can be taken further, because finding the first word sense is in some ways equivalent to finding the first translation, which is especially important for cross-lingual information retrieval in the same areas where named entity recognition is important. A method which significantly outperforms previously known comparable methods for finding translations of named entities in a cross-lingual setting has been proposed by the author (Lindén, 2004, 2005 forthcoming).

As Kilgarriff (2003b) points out, automatically identifying a word's senses has been a goal since the early days of computational linguistics, but is not one where there has been resounding success. He suggests that the underlying problem may be unclarity as to what a word sense is (Kilgarriff, 1997). A word might not have been seen in a context because it is not acceptable there, or it might not have been seen there simply because the corpus was not big enough (Kilgarriff, 2003b). In the following, we will first look at the frequency aspect and then at the acceptability aspect.

---

from an educated guess."

24.4.1 Frequency Distribution

Where a lexicographer is confronted with a large quantity of corpus data for a word, then, even if all of the examples are in the same area of meaning, it becomes tempting to allocate the word more column inches and more meanings, the lexicographer Kilgarriff admits in (Kilgarriff, 2004) and considers the words *generous* and *pike* as examples:

> *Generous* is a common word with meanings ranging from generous people (who give lots of money) to generous helpings (large) to generous dispositions (inclinations to be kind and helpful). There are no sharp edges between the meanings, and they vary across a range. Given the frequency of the word, it seems appropriate to allocate more than one meaning, as do all of the range of dictionaries inspected. *Pike* is less common (190 BNC occurrences, as against 1144) but it must be assigned distinct meanings for fish and weapon (and possibly also for Northern English hill, and turnpike, depending on dictionary size), however rare any of these meanings might be, since they cannot be assimilated as minor variants. Pike-style polysemy, with unassimilable meanings, is the kind that is modeled in this paper. Where there is generous-style ambiguity, one might expect less skewed distributions, since the lexicographer will only create a distinct sense for the 'generous disposition' reading if it is fairly common; if the lexicographer encounters only one or two instances, they will not. Polysemy and frequency are entangled.

In the same article, Kilgarriff (2004) observes that the dominance of the most common sense increases with n, the frequency of the word. In additional corpus data, we find additional senses for words. Since a majority of the words are monosemous[2], finding additional senses for them dominates the statistic. On the average, the proportion of the dominant sense therefore increases with n simply because the proportion of the first sense, $(n-1)/n$, compared to that of the additional sense, $1/n$, increases with n. He proceeds to demonstrate that the distribution of word senses roughly follows a Zipfian power-law similar to the well-known type/token distribution (Baayen, 2001, Zipf, 1935). Kilgarriff uses the sense-tagged SemCor database (Mihalcea, 2004) for empirical figures on the proportion of the most common sense for words at various frequencies, and compares the empirical figures with the figures his model predicts when initialized with the word frequency distribution from the British National Corpus (BNC) (Burnard, 1995). The fit between the SemCor and the predicted figures makes it believable that word frequencies and word sense frequencies have roughly similar distribu-

[2]WordNet is an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets. WordNet contains approximately 126,000 monosemous words with as many word senses, and 26,000 polysemous words with 78,000 word senses (Miller et al., 2003).

tions and that we can expect the skew to become more pronounced for higher values of n.

The conclusions we can draw from Kilgarriff (2004) are that a large-scale domain-independent word sense disambiguation system, which always chooses the most common sense out of two or more senses, will over time perform accurately in 66–77 % of the ambiguous cases based on the weighted average of the SemCor figures, or even in 66–86 % of the cases according to the figures predicted by the larger BNC corpus model. For high-frequency words, the ambition of a lexicographer to account for all the source material rather than for all the senses is a partial explanation for why some word senses are difficult to disambiguate even for humans. If such senses were disregarded, the higher predicted proportions of the dominant sense may in fact be more valid for the high-frequency words. Another implication of the Zipfian distribution is that over time all words are likely to appear in most contexts with a very low probability, and in practice most word senses will never have been seen more than once in any specific context.

24.4.2 Acceptability in Context

As soon as we start limiting the acceptability of words in certain contexts, we begin losing creative language use. One possibility is to relate the contents of a sentence to the world we live in, in order to estimate the plausibility of the sentence. However, this will complicate matters, because we then also have to model the plausibility of events in the world. An approximation of how objects and events of the world relate to one another is provided by an ontology. Unfortunately, there is yet no world-wide ontology around, but we have fairly large thesauri.

The difference between a thesaurus and an ontology is that the former deals with words and their relations observable in language use and the latter deals with objects and their relations in the world we live in. To high-light the distinction, we can consider the famous quote “Colorless green ideas sleep furiously” by Chomsky (1957). From a purely language use perspective this full sentence is unexpectedly likely occurring more than 5,700 times on the world-wide web. It is so common that it can be regarded as idiomatic. From an ontological perspective, the fact that it has been repeated into idiomhood by the world’s linguists does not make its content more plausible. Compositionally it still means little, but contextually it is a very pregnant construction. However, people tend to speak and write more often about things they have or would like to have experienced than they spend time producing and repeating random sequences of words, so the language we can observe is a noisy reflection of the relations between objects in the world. As a consequence, the difference is not so wide between a thesaurus constructed from observations of language use and an ontology constructed from observations of the world.

A bigger practical problem is that thesauri usually do not contain well-defined word senses that we could use for plausibility judgments. In an effort to clarify the relation between words and their multiple meanings Kilgarriff (2003a) tries to explain why thesauri do not really contain word senses. The first priority of authors of thesauri is to give coherent meaning-clusters, which results in quite different analyses from those in dictionaries, where the first priority is to give a coherent analysis of a word in its different senses (Kilgarriff and Yallop, 2000). From a practical point of view, if we wish to use a thesaurus for a natural language processing (NLP) task, then, if we view the thesaurus as a classification of word senses, we have introduced a large measure of hard-to-resolve ambiguity to our task (Kilgarriff, 2003a). For this reason Kilgarriff claims that, even though Roget may have considered his thesaurus (Roget, 1987) a simple taxonomy of senses, it is better viewed as a multiple-inheritance taxonomy of words.

The direct consequence of Kilgarriff's argument is that a thesaurus is perhaps useful as a backbone for a generative lexicon, but as such the words in a thesaurus are ambiguous. Kilgarriff's argument is easier to understand if we keep in mind that the meaning of a word is defined by the contexts in which it occurs. The real problem is that a meaning-cluster in a thesaurus seldom includes the common contexts in which the words of the meaning-cluster occur. So what can we use a thesaurus for? Systems which try to discover word senses, also classify words based on their context into maximally coherent meaning-clusters, i.e., thesauri can serve as test beds for automatic word sense discovery systems. The somber consequence of Kilgarriff's argument is that for NLP systems the words in a meaning-cluster are in fact an epiphenomenon[3]. The valuable part is the context description by which the words were grouped. The context description is a compact definition of the meaning of the word cluster and this is the part that is usually made explicit in a regular dictionary analyzing the senses of a word. It is the context description that can be used for determining the acceptability of the word sense in various contexts.

24.5 Word Sense Dictionary Specification

If we use a generative lexicon to determine the acceptability of a word sense in context and the lexicon provides hard constraints, we will end up not covering creative language use after all. We could, however, account for creative lan-

[3]This is not to say that word sense and thesaurus discovery efforts are futile. Word lists are primarily intended for consumption by systems that are capable of filling in the appropriate context descriptions themselves, e.g., human beings. A central issue in information retrieval (IR) research is to devise strategies which cope with missing context. This may partially explain why IR often seems to have more to offer thesaurus makers than the other way around, see (Sanderson, 2000).

guage use by basing plausibility judgments[4] on observable language. Ideally, a lexicon provides structure and soft constraints based on context descriptions giving more plausibility to more likely objects and events.

To summarize the discussion of the previous sections, we can set up a general wish list of what a context description of a word sense in an ideal lexicon should contain, loosely based on the idea of a generative lexicon (Pustejovsky, 1998): *part of speech* categories, *argument structure* of arguments and adjuncts, *event structure* for the argument structure, *qualia structure* describing an object, its parts, the purpose and the origin of the object, *interlexical relations*, e.g., synonymy, antonymy, hyponymy, entailment, translation, *plausibility estimate* by providing all of the above with frequency or probability information[5].

An example of the plausibility information the lexical model needs to incorporate is given by Lapata and Brew (2004), where they highlight the importance of a good prior for lexical semantic tagging. They find a prior distribution for verb classes based on Levin (1993), and they obtain their priors directly from subcategorization evidence in a parsed but semantically untagged corpus.

Another example is the prevalence ranking for word senses according to domain, which should be included in the generative lexical look-up procedure. The sense distributions of many words depend on the domain. Giving low probability to senses that are rare in a specific domain permits a generic resource such as WordNet to be tailored to the domain. McCarthy et al. (2004) present a method which calculates such prior distributions over word senses from parsed but semantically untagged corpora.

24.6 Conclusion

In text we can observe word forms which through morphological analysis get a base form. A base form may have several meanings which together form a lexeme. An explicit *meaning–base form* pair, i.e., a word sense, is an artifact we cannot observe directly. We can only observe word usages. The only evidence we have for a word sense is found in a dictionary via the definitions and glosses provided by a lexicographer reflecting meaningful groups of word usages.

[4] A plausibility judgment is at least a weak partial ordering of the relative plausibility of statements.

[5] From a Bayesian statistics point of view we would have prior linguistic information combined with the posterior information provided by corpus data. Before we have seen any data, our prior opinions about what the true relationships might be can be expressed in a probability distribution over the feature structure weights that define the relationships. After we look at the corpus data (or after our lexicon is adapted to the data), our revised opinions are captured by a posterior distribution over the feature structure weights.

We have briefly described the criteria lexicographers use when they decide which word usages constitute a word sense. The fact that the bulk of all language use is a reflection of the world we live in, makes some word senses of a word dominant. Most previously unseen word usages are creative simply because they are unexpected or surprising at the time. A natural language processing (NLP) system needs to recognize that a usage is unexpected. However, the context in which the usage appears is what the word means and should be recorded for future reference, e.g., telephones used to be stationary until the advent of mobile phones, so a sentence like "He walked down the street talking on the phone" was implausible 30 years ago, but is now highly likely and the walking-talking context has become part of the meaning of a telephone.

We have argued that word meaning is not discrete. However, the meaning of words is quantized into word senses in a dictionary. If we need a common world view, we can refer to a sense inventory of an agreed upon dictionary, otherwise we can as well compare word contexts directly.

References

Baayen, Harald R. 2001. *Word Frequency Distributions*, vol. 18 of *Text, Speech and Language Technology*. Dordrecht: Kluwer Academic Publishers.

Burnard, Lou. 1995. *Users' Reference Guide for the British National Corpus, version 1.0*. Oxford University Computing Services, Oxford, UK.

Chomsky, Noam. 1957. *Syntactic structures*. Janua Linguarum Series Minor, Volume 4. The Hague, The Netherlands: Mouton de Gruyter.

Gleitman, Lila R. 1990. The structural sources of verb meaning. *Language Acquisition* 1:3–55.

Gleitman, Lila R. 2002. Verbs of a feather flock together II: The child's discovery of words and their meanings. In B. E. Nevin and S. B. Johnson, eds., *The Legacy of Zellig Harris: Language and information into the 21st century*, vol. 1: Philosophy of science, syntax and semantics of *Current Issues in Linguistic Theory*, pages 209–229. John Benjamins Publishing Company.

Harris, Zellig S. 1954. Distributional structure. *Word* 10:146–162.

Harris, Zellig S. 1968. Mathematical structures of language. *Interscience Tracts in Pure and Applied Mathematics* 21(ix):230 pp.

Kaplan, Frédéric. 2001. *La Naissance d'une Langue chez les Robots*. Collection Technologies et Culture. Paris, France: Hermes Science Publications.

Kilgarriff, Adam. 1997. I don't believe in word senses. *Computers and the Humanities* 31(2):91–113.

Kilgarriff, Adam. 2001. Generative lexicon meets corpus data: the case of non-standard word uses. In P. Bouillon and F. Busa, eds., *The Language of Word Meaning*, pages 312–328. Cambridge: Cambridge University Press.

Kilgarriff, Adam. 2003a. Thesauruses for natural language processing. In *Proceedings of the 2003 International Conference on Natural Language Processing and Knowledge Engineering*. Beijing: Beijing Media Center.

Kilgarriff, Adam. 2003b. What computers can and cannot do for lexicography, or Us precision, them recall. Tech. Rep. ITRI-03-16, Information Technology Research Institute, University of Brighton. Also published in Proceedings of ASIALEX.

Kilgarriff, Adam. 2004. How dominant is the commonest sense of a word? In P. Sojka, I. Kopeček, and K. Pala, eds., *Proceedings of TSD 2004, Text, Speech and Dialogue 7th International Conference*, vol. 2448 of *LNAI*, pages 1–9. Brno, Czech Republic: Springer-Verlag, Berlin.

Kilgarriff, Adam and Colin Yallop. 2000. What's in a thesaurus? In *Proceedings of LREC 2000, the 2nd International Conference on Language Resources and Evaluation*, pages 1371–1379. Athens.

Lapata, Mirella and Chris Brew. 2004. Verb class disambiguation using informative priors. *Computational Linguistics* 30(1):45–75.

Levin, Beth. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. Chicago and London: University of Chicago Press.

Lindén, Krister. 2004. Finding cross-lingual spelling variants. In *Proceedings of SPIRE 2004, the 11th Symposium on String Processing and Information Retrieval*. Padua, Italy.

Lindén, Krister. 2005 forthcoming. Multi-lingual modeling of cross-lingual spelling variants. *Information Retrieval* .

McCarthy, Diana, Rob Koeling, Julie Weeds, and John Carroll. 2004. Automatic identification of infrequent word senses. In *Proceedings of the 20th International Conference of Computational Linguistics, COLING-2004*, pages 1220–1226. Geneva, Switzerland.

Mihalcea, Rada. 2004. Software and data sets – semcor. [http://www.cs.unt.edu/ rada/downloads.html#semcor].

Miller, George A., Christiane Fellbaum, Randee Tengi, Susanne Wolff, Pamela Wakefield, Helen Langone, and Benjamin Haskell. 2003. Wordnet – a lexical database for the English language. [http://www.cogsci.princeton.edu/∼wn/index.shtml].

Pustejovsky, James. 1998. *The Generative Lexicon*. The MIT Press.

Resnik, Philip and David Yarowsky. 2000. Distinguishing systems and distinguishing senses: New evaluation methods for word sense disambiguation. *Natural Language Engineering* 5(3):113–133.

Roget, Peter Mark. 1987. *Roget's Thesaurus*. Longman, Longman Edition edit by Betty Kirkpatrick edn. Original edition 1852.

Sanderson, Mark. 2000. Retrieving with good sense. *Information Retrieval* 2(1):49–69.

Wittgenstein, Ludwig. 1953. *Philosophical Investigations*. Oxford: Basil Blackwell. Translated by G. E. M. Anscombe.

Zipf, George Kingsley. 1935. *The Psycho-biology of Language: An Introduction to Dynamic Philology*. Boston, USA: Houghton Mifflin.

25

Exploring Morphologically Analysed Text Material

MIKKO LOUNELA

In text linguistics, it is possible to carry out research by carefully analysing a small set of texts, sometimes just a few. For some examples of this kind of text analysis, see Heikkinen (2005). Such a methodological choice is not easily combined with the idea of using corpus-based methods and quantitative analysis as an essential part of research. In text linguistics, however, the text type constitutes an important research problem, and some work has been done in classifying texts according to their quantitative morphological and syntactic characteristics. Such work has been going on for a few decades now, see e.g. Biber (1988). For a related approach to Finnish texts, see Saukkonen (2001).

In the Research Institute for the Languages of Finland (RILF), we are aiming at a fruitful combination of quantitative morpho-syntactic analysis and deep text analysis based on Lexical Functional Grammar, i.e. LFG (2004). This work includes providing a morpho-syntactic analysis (in due form) of a selected group of texts, and calculating a "morphological fingerprint" of the text group. One group of texts forms a text material, usually of moderate size (consisting of fewer than 100 texts, with approximately 100,000 words). This article focuses on the problems and choices in adding the morpho-syntactic annotation to the text material, and in defining intuitive linguistic categories such as part-of-speech, verb, finite verb, and tense using semi-automatic word-level morpho-syntactic analysis.

The language of our texts is Finnish. The design of our materials is based on the XML (1996-2004) language, using modified TEI (2001-2003) P4 structure. The morphological annotation is based on the analysis provided

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

by FINTWOL (1995-2000), a morphological analyser developed at Lingsoft (version 1998/03/02), based on the Two-level model introduced by Koskenniemi (1983). An overview of FINTWOL's tag set is presented in Fintwol/tags (2001). The morphological analysis goes through a careful hand-made disambiguation and augmentation. Our model for text materials is described in Lehtinen and Lounela (2004). The exploration of the text material is carried out using the Xquery (2000-2005) language.

25.1 Morphological Analysis and Text Structure

The FINTWOL morphological analyser provides each word of the material with morphological information. This information includes the base form (lemma) of the word, and an unordered set of tags, expressing morphological features of the word. If the word can represent more than one word-form, FINTWOL will list all its possible readings. In the case of compound words, the word-internal boundaries are marked in the lemma. At RILF, we use our own pre-processor to enhance FINTWOL's capabilities in processing Finnish abbreviations and numerical expressions.

The following illustrates FINTWOL analysis and the ambiguity it may produce. The Finnish word-form *alustamassa* may be interpreted either as a compound noun it alustamassa ("platform mass"), or a third infinitive or a deverbalised derivation of the verb *alustaa*, ("knead" or "format"):

```
"<alustamassa>"
      "alusta#massa"  N NOM SG
      "alustaa"  V INF3 INE
      "alustaa"  DV-MA INE SG
```

In order that the FINTWOL analysis would be usable in the TEI-format, the information it gives has to be split and embedded in the XML-element. The element for a text word in TEI P4 definition is "w", and it has such attributes as "lemma" for the base-form and "type" for the part-of-speech information. Following the Corpus Encoding Standard, i.e. CES (2000), we have added an attribute, "msd", to include the morpho-syntactic description in the word element. The following (simplified) example shows how the Fintwol analysis is embedded in XML-encoding:

```
<w lemma="alusta#massa" type="N" msd="NOM SG">alustamassa</w>
<w lemma="alustaa" type="V" msd="INF3 INE">alustamassa</w>
<w lemma="alustaa" type="DV-MA" msd="INE SG">alustamassa</w>
```

The XML-type word elements are then disambiguated by hand (only the most likely analysis is retained), and some information concerning multi-word features (e.g., perfect tense) is added. The words are then included in the general text structure, e.g., in text chapters, headers, legends, etc. This type of text material can be analysed quantitatively according to its morpho-syntactic

features.

25.2 Quantitative Lexical Analysis

The morpho-syntactic fingerprint of a group of texts consists mainly of figures and frequency lists of the morpho-syntactic features of the words in the material. At present, the fingerprint that we have designed in RILF consists of four different parts: (1) the general part, (2) the verbal part, (3) the nominal part, and (4) the lexical part.

The general part includes information such as the average lengths of texts, sentences and clauses, and the frequencies of punctuation marks, lemmas, and most common word-forms and parts-of-speech in the material. The nominal part concerns words of the types "N" (noun), "A" (adjective), "PRON" (pronoun) and "NUM" (numeral). It includes the frequencies of cases, comparatives, numera, word-forms and lemmas as well as the frequency lists of the most common word-forms and parts-of-speech of the nominals in the material. The verbal part of the fingerprint includes the frequencies of features such as voice, mood and tense as well as frequencies of infinitive forms, participles and the most common verbal lemmas and word forms. The lexical part of the fingerprint consists of frequency lists of the most common lemmas and word-forms of each of the parts-of-speech (values of the "type"-attribute) in the material.

Some of these features can be obtained directly from the FINTWOL analysis, while some of them require combining the FINTWOL tags and interpreting the combinations. In the remainder of this article I will consider defining more or less problematic features such as part-of-speech, verb, finite verb and tense. A sample verbal fingerprint analysis along with the xquery code used to produce it can be seen on the web site of RILF, at <http://www.kotus.fi/julkaisut/2005-ml-1/>.

25.3 Part-of-speech

The transformation from FINTWOL to XML includes recognising the part-of-speech tag in the FINTWOL analysis. In the previous example, the most obvious candidate is the first tag of the analysis, but this is not always the case. In the following FINTWOL analysis of the word-form *kaavoittaja* ("planner"), the obvious part-of-speech tag "N" (noun) is preceded by the tag providing information about the derivation of the word-form ("DV-JA"):

```
"<kaavoittaja>"
      "kaavoittaja"  DV-JA N NOM SG
```

There are also FINTWOL analyses where no obvious part-of-speech tag is present, and those in which we have to choose between more than one good candidate. In the last line of the first example (*alustamassa*), the best

candidate for the part-of-speech is in this particular case the derivation tag "DV-MA", as other possibilities are in practice less appropriate. More about the part-of-speech problematics concerning current morphological analysers for Finnish can be found in Heikkinen and Lounela (forthcoming).

At RILF, we have developed a simple algorithm for automatically finding the best part-of-speech candidate in the FINTWOL analysis. The algorithm divides the FINTWOL tags into four classes, of which we choose the most likely part-of-speech in the following manner.

1. If the analysis contains one or more of the tags "A", "ABBR", "AD-A", "ADV", "C", "INTJ", "N", "NUM", "PP", "PREP", "PRON", "PSP", or "V", choose the one that appears last in the tag sequence.
2. If the analysis does not contain any of the tags mentioned above, choose the last of "Q", "PCP1", "PCP2", or "A/N".
3. If the analysis does not contain any of the tags mentioned above, choose the last tag indicating the derivative properties of the words (any tag beginning "D?-", where "?" denotes any character).
4. If none of the above applies, choose the first tag in the analysis.

This algorithm gives us the following list of part-of-speech tags, when applied to a 20,000-word sample of material from the Finnish newspaper Aamulehti after analysis by FINTWOL and without any subsequent disambiguation.

TAG	PART-OF-SPEECH
UNKNOWN	Unrecognised word-form
A	Adjective
A/N	Adjective or noun
ABBR	Abbreviation
AD-A	Ad-adjective
ADV	Adverb
C	Conjunction
DV-MA	Deverbal derivation with ending "ma"
FORGN	Foreign word
INTJ	Interjection
N	Noun
NUM	Numeral
PCP1	First participle
PCP2	Second participle
PP	Post- or preposition
PRON	Pronoun
PSP	Postposition
V	Verb

25.4 Verb

As the part-of-speech has its FINTWOL-based operational definition, we can start to define other linguistic features. Here, I will focus on the verbs and the morpho-syntactic properties that are closely related to them.

25.4.1 Verb

The definition of the verb itself might seem quite unproblematic, since it is a primary part-of-speech category, as seen earlier. However, when we take a look at analyses of a few text samples, the picture changes. When analysing verb chains, such as those containing negation *ei juossut*, "[he/she/it] did not run", or the perfect tense, *on juossut*, "[he/she/it] has run", we notice that the number of the finite verb forms in the negative construction is two, while the perfect construction has only one finite verb form, as the *juossut* is defined as a participle form in the construction. The infinitive form *saamme juosta* "[we] may run" consists, again, of two verbs. The analyses for the verb form *juossut* in the example have been selected from the three alternatives given by the FINTWOL analyser, the third interpretation being an adjective. The analysis for the form *juosta* is selected from two analyses, the other possibility being present negative passive. All the following examples will be manually disambiguated:

```
<w lemma="ei" type="V" msd="NEGV SG3">ei</w>
<w lemma="juosta" type="V" msd="PAST ACT NEG SG">juossut</w>

<w lemma="olla" type="V" msd="COP PRES ACT SG3">on</w>
<w lemma="juosta" type="PCP2" msd="ACT POS NOM SG">juossut</w>

<w lemma="saada" type="V" msd="PRES ACT PL1">saamme</w>
<w lemma="juosta" type="V" msd="INF1 NOM">juosta</w>
```

When counting verbs, we divide the verb category into two: (1) semantic verbs and (2) grammatical verbs. The semantic verbs include the participle forms of the temporal verb chains forming perfects and pluperfects, as well as all the words of the type "V", except for the auxiliaries in the negative and temporal verb constructions.

The grammatical verbs include the same set of words as the semantic verbs, with some exceptions. The infinitive verb forms (marked with "INF1", "INF2", "INF3", etc.) are excluded, and the auxiliary is selected from the temporal chains. In the temporal chains, the participle has a part-of-speech marker "PCP2", so this can be achieved by just counting the "V"-tags, and excluding the infinitives.

To make the operational definitions for all the possible tenses, we have to mark the perfect and the pluperfect tenses in the material. An extra attribute ("function") is added to the data model for this purpose.

```
<w ... type="V" msd="COP PRES ACT SG3" function="P">on</w>
<w ... type="PCP2" msd="ACT POS NOM SG" function="P">juossut</w>
```

For the sake of consistency, it would probably be necessary to function-mark modal verb chains, such as *saamme juosta* (see above), but this is not done at present.

25.4.2 Finite Verb and Tense

According to the latest authoritative and quite comprehensive grammar of Finnish, *Iso Suomen Kielioppi*, by Hakulinen et al. (2005), a finite verb in Finnish is a verb that is inflected in tense, mood and person. A finite verb functions as the nucleus of a clause. Identifying the finite verbs is essential for obtaining figures related to clauses, which we consider very important. Mapping this definition of finiteness to FINTWOL analysis is, however, problematic, for at least two reasons.

Firstly, in the FINTWOL analysis the indicative mood is provided as the default value for all verb forms. In order to follow the definition given in Hakulinen et al. (2005), we should know which verbs inflect in mood, in order to to be able to identify the finite verbs. This information is, however, not available.

Second, while the FINTWOL analysis does not include a tag for person inflection in the negative verb forms, it includes one in the negative auxiliaries in negative verb chains, eg. *emme juokse* ("[we] do not run"), below. This means that in negative forms finiteness is divided between the semantic verb and the negative auxiliary. Identifying it would require information about word dependencies, but that kind of information is not available in the FINTWOL analysis:

```
<w lemma="ei" type="V" msd="NEGV PL1">emme</w>
<w lemma="juosta" type="V" msd="PRES ACT NEG">juokse</w>
```

At present, the fingerprint analysis defines finite verbs as a set of semantic verbs, where the active or passive marker is present, with the infinitive forms excluded.

We use the number of finite verbs as an indicator of the number of clauses in the text materials. Concerning the problems and uses of this type of work, see Heikkinen et al. (2000). The following sentence (*älä juokse ja huuda*, "do not run and shout") is interpreted as having two finite verbs, and thus clauses, even though it has only one word with inflection markings for person, none for tense, and three for mood:

```
<w lemma="ei" type="V" msd="NEGV IMPV ACT SG2">Älä</w>
<w lemma="juosta" type="V" msd="IMPV ACT NEG SG">juokse</w>
<w lemma="ja" type="C" msd="COORD">ja</w>
<w lemma="huutaa" type="V" msd="IMPV ACT NEG SG">huuda</w>
<w lemma="." type="PUNCT" msd="FULLSTOP">.</w>
```

Having all the above definitions, defining the tenses of the verbs is quite straightforward. We use the finite verbs as the base set of words expressing temporal information of the texts. As the perfect and pluperfect are explicity marked, we can identify the tenses directly, using the FINTWOL tags "PAST" and "PRES" combined with the information provided by the function attribute.

25.5 Conclusion

In this article, I have presented proposals for operational definitions for some linguistic categories for Finnish. The proposals are based on hand-augmented morphological analysis of Finnish texts, the analysis being provided by the FINTWOL morphological analyser. The defined categories include part-of-speech, verb, finite verb, and tense.

1. Part-of-speech marker can be selected from the FINTWOL analysis as being
   (a) the last tag indicating primary word category (adjective, abbreviation, ad-adjective, adverb, conjunction, interjection, noun, numeral, post/preposition, pronoun, postposition, or verb), or
   (b) the last tag indicating secondary word category (quantifier, first or second participle, or adjective/noun), if the above does not apply, or
   (c) the last tag indicating derivative information, if none of the above applies, or
   (d) the first tag of the analysis, if none of the above applies.
2. A semantic verb is a word of part-of-speech "V", with the temporal and negative auxiliaries excluded, and with the participle forms ("PCP2") of the perfective and pluperfective verb chains included.
3. A grammatical verb is a word of part-of-speech "V", with the infinitive verb forms excluded.
4. A finite verb is a semantic verb with voice, active ("ACT"), or passive ("PSS"), with the infinitive forms excluded.
5. The tenses are counted on the basis of finite verbs. The markers "PRES" and "PAST" indicate present and past tense, and a special attribute ("function", with values "P" for perfect and "PL" for pluperfect) is added to words in the temporal verb chains to indicate corresponding tenses.

The overall process of giving functional morphological definitions to the general grammatical categories raises some issues concerning the design principles of morphological analysers for the Finnish language. Taking these things into account would greatly enhance the usability of such analysers.

First, the major linguistic categories, such as part-of-speech, should be consistently included in the analysis of each word. Second, no categories should be left as the default, such as the indicative mood is in the present FINTWOL analysis. The fundamental set may not be clear, which makes deducing the set of the words with the default value hard, or even impossible. Third, the ordering of the tags does matter. Tabular or XML-style representation of the analysis would help the user to identify the features behind the markers, and to see what information may be missing. Finally, a complete documentation of all the categories and markers used by the analyser is essential for its scientific use.

Acknowledgments

The views expressed in this article are based on practical work on annotating, analyzing and exploring text materials at the Research Institute for the Languages of Finland in 2000-2005. Part of the work was funded by the Academy of Finland in 2003-2004. I am grateful to Vesa Heikkinen and Outi Lehtinen for fruitful cooperation. I would also like to thank Vesa Heikkinen for his comments regarding this article.

References

Biber, Douglas. 1988. *Variation across speech and writing*. Cambridge: Cambridge University Press.

CES. 2000. Corpus encoding standard. <http://www.cs.vassar.edu/CES/>. Vassar College. Visited March 2005.

FINTWOL. 1995-2000. Fintwol. <http://www.lingsoft.fi/cgi-bin/fintwol/>. Lingsoft, inc. Visited March 2005.

Fintwol/tags. 2001. Tags (partial list). <http://www.lingsoft.fi/doc/fintwol/intro/tags.html>. Lingsoft, inc. Visited March 2005.

Hakulinen, Auli, Maria Vilkuna, Riitta Korhonen, Vesa Koivisto, Tarja Heinonen, and Irja Alho. 2005. *Iso suomen kielioppi*. Helsinki: SKS.

Heikkinen, Vesa, ed. 2005. *Tekstien arki*. Helsinki: Gaudeamus.

Heikkinen, Vesa, Outi Lehtinen, and Mikko Lounela. 2000. Ihminen ja kone tekstiä mankeloimassa, kuusikohtauksinen keskustelu. In H. Sulkala and L. Nissilä, eds., *XXVII Kielitieteen päivät Oulussa 19. - 20.5.2000*. Oulu: University of Oulu.

Heikkinen, Vesa and Mikko Lounela. forthcoming. Sanaluokka morfologisen analyysin kategoriana. In K. Kerge and M.-M. Sepper, eds., *Finest Linguistics. Proceedings of the Annual Finnish and Estonian Conference of Linguistics. Tallinn, May 6-7, 2004*. Tallinn: TPÜ.

Koskenniemi, Kimmo. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Helsinki: University of Helsinki, Department of General Linguistics.

Lehtinen, Outi and Mikko Lounela. 2004. A model for composing and (re-)using text materials for linguistic research. In M. Nenonen, ed., *Papers from the 30th Finnish Conference of Linguistics*. Joensuu: University of Joensuu.

LFG. 2004. Lexical Functional Grammar (Stanford Web Site). <http://www-lfg.stanford.edu/lfg/>. University of Stanford. Visited March 2005.

Saukkonen, Pauli. 2001. *Maailman hahmottaminen teksteinä*. Helsinki: Helsinki University Press.

TEI. 2001-2003. Text encoding initiative. <http://www.tei-c.org/>. TEI Consortium. Visited March 2005.

XML. 1996-2004. Extensible markup language (xml). <http://www.w3.org/XML/>. World Wide Web Consortium. Visited March 2005.

Xquery. 2000-2005. Xml query (xquery). <http://www.w3.org/XML/Query/>. World Wide Web Consortium. Visited March 2005.

26

Information Structure and Minimal Recursion Semantics

GRAHAM WILCOCK

26.1 Introduction

Comparing English and Finnish, and simplifying a complex issue very much, we can say that English has fixed word order and Finnish has free word order. Syntactic theories such as HPSG (Sag and Wasow, 1999) have provided relatively successful descriptions of English, using a phrase structure approach to capture generalizations about fixed word order. Software tools such as LKB (Copestake, 2000) have been developed and made freely available to provide good support for implementing these descriptions.

Free word order in Finnish is described in depth by Vilkuna (1989), both in terms of syntax and its discourse functions. Theories such as HPSG have been much less successful in providing descriptions of languages such as Finnish, where discourse functions play a major role in word order. One of the problems in HPSG is that its account of information structure and discourse functions has not yet been sufficiently developed. This paper[1] addresses one aspect of this issue, namely what kind of representation is appropriate for information structure in HPSG. Another paper in this volume (Jokinen, 2005) presents an implementation of Finnish discourse syntax in an HPSG framework using LKB.

Sections 26.2 and 26.3 describe two different approaches to representing information structure: a syntax-oriented approach which has been proposed

[1] An earlier version of this paper (Wilcock, 2001) was presented at the 13th Nordic Conference on Computational Linguistics, Uppsala, 2001.

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

in HPSG, and a semantics-oriented approach which has been used in a practical dialogue system. In both cases we note the problem of representing focus scope. Section 26.4 briefly compares the functional approach taken in Systemic Functional Grammar.

Section 26.5 describes the Minimal Recursion Semantics (MRS) representation developed for HPSG, and shows how quantifier scope is handled in MRS. Section 26.6 proposes a way to extend MRS to include information structure. We raise the question whether focus scope can be handled in MRS in a similar way to quantifier scope, and we show how a wide range of focus scope examples can be treated in the extended MRS representation.

26.2 Information Structure: A Syntactic Approach

A representation for information structure in HPSG was proposed by Engdahl and Vallduví (1996). Arguing that information structure is a distinct dimension, which should not be associated only with phonology, only with syntax, or only with semantics, they propose that a feature INFO-STRUCT should be located within the CONTEXT[2] feature in the HPSG framework, rather than in CATEGORY (syntax) or CONTENT (semantics). INFO-STRUCT includes FOCUS and GROUND, the latter including LINK and TAIL.

However, the specific representation which they use is syntactic: LINK and FOCUS are equated with the syntactic constituents (NPs and VPs) which realize the topic concept and the focus information. As the primary concern of Engdahl and Vallduví (1996) is with information *packaging*, this has the advantage of facilitating the description of the realization of information structure (by intonation in English, by word order in Catalan), but it has the major disadvantage that the packaging is only indirectly tied to the information which is packaged, which is itself part of the semantic content. In a footnote, Engdahl and Vallduví themselves suggest that it would be more appropriate for the value of INFO-STRUCT to be structure-shared with the CONTENT information.

26.2.1 Focus Scope in a Syntactic Approach

This syntax-based representation of information structure enables the distinction between narrow focus and wide focus to be represented. Engdahl and Vallduví give the example *The president hates the Delft china set* which can be interpreted either with narrow focus on the object noun phrase (26.1) or with wide focus on the whole verb phrase (26.2).

(26.1) The president hates [F the Delft china set].

(26.2) The president [F hates the Delft china set].

[2] There are a number of issues concerning the role of the CONTEXT feature in HPSG. Some of them are discussed by Wilcock (1999).

To represent these alternatives, the value of FOCUS at higher nodes (S and VP) is equated with the smaller syntactic constituent (the object NP) to represent the narrow focus reading, or with the larger syntactic constituent (the whole VP) to represent the wide focus reading, as shown by examples (17) and (18) of Engdahl and Vallduví (1996).

This would be an elegant way to capture the narrow and wide focus readings. However, there are a number of cases where informational partitioning does not correspond to syntactic constituency. Among the examples given by Engdahl & Vallduví are subject-verb focus (26.3) and complex focus (26.4):

(26.3) What happened to the china set? [F The BUTLER BROKE] the set.

(26.4) Who did your friends introduce to whom?
John introduced BILL to SUE, and **Mike** introduced . . .

To handle these examples, Engdahl & Vallduví change the representation so that set values will be used: the value of FOCUS will not be a single syntactic constituent which exactly spans the focus scope, but an otherwise arbitrary set of syntactic constituents which together make up the relevant sequence of words. The representation thereby loses its initial elegance. With this change, Examples 26.1 and 26.2 will have a singleton set value for FOCUS, and set values will also be used for LINK and TAIL.

26.2.2 HPSG vs. CCG

Despite adopting a syntax-oriented representation, Engdahl and Vallduví (1996) argue that information structure is a distinct dimension, and locate INFO-STRUCT in the HPSG CONTEXT feature.

Steedman (1991) argues that there is a systematic correspondence between information structure, intonation and syntactic constituency, and it is a strength of Combinatory Categorial Grammar (CCG) that it allows suitable syntactic constituents which support this correspondence.[3] Engdahl and Vallduví (1996) argue that there is no such correspondence between information structure and syntactic constituency, and that it is a strength of HPSG's multidimensional representation that we are not forced to assume any such correspondence. Both approaches could be said to over-emphasise the role of syntax, in an area where semantics and pragmatics should be more central.

26.3 Information Structure: A Semantic Approach

We now examine a different approach to information structure, based on the practical requirements of dialogue modelling in robust dialogue system projects. These requirements appear to support a closer link between the information structure representation and the semantic representation. Dialogue

[3]Related problems in using HPSG for incremental generation, compared with CCG, are discussed by Wilcock (1998).

responses need to be generated from the semantic information. Old and new discourse referents need to be distinguished, and referents are usually identified by indices in the semantic representation. In addition, topic continuities and topic shifts need to be tracked, and the topics are also identified by semantic indices, even when a topic is some kind of event.

As an example of this approach we take the dialogue modelling framework used in PLUS (Pragmatics-based Language Understanding System), described by Jokinen (1994). In PLUS, the semantic representation consists of flat quasi-logical forms with simple indices for discourse referents. The dialogue manager component takes account of information structure and decides what semantic representations to supply to the generator. Jokinen defines **Topic** as a distinguished discourse entity which is talked about, and which is an instantiated World Model concept. **NewInfo** is a concept or property value which is *new* with respect to some Topic. The representation for both is based directly on the semantic representation. Jokinen gives an example from PLUS (Topics are in italics, NewInfo bold-faced):

(26.5) User: *I need a car*.
System: Do you want to **buy or rent** *one*?
User: **Rent**. (topic: *car*)
System: **Where**? (topic: *rent*)
User: In **Bolton**. (topic: *rent*)
...

Jokinen (1994) explains that in the first system contribution in (26.5), NewInfo is the disjunction 'buy or rent', which has the representation:

(26.6) Goal: know(s,[wantEvent(w,u,d),disj(d,b,r),
buyEvent(b,u,c,_),hireEvent(r,u,c,_),car(c),user(u)])
NewInfo: disj(d,b,r)

Compared with the syntax-oriented representation of information structure, this semantics-oriented representation appears to have the advantage of facilitating topic tracking and distinguishing old and new referents, due to the direct use of semantic indices (c = *car*, r = *rent*, etc.). Further examples of its use in practical dialogue modelling are described by Jokinen (1994). In the PLUS system, a pragmatics-based Dialogue Manager explicitly manages information structure. Response planning in the Dialogue Manager always starts from NewInfo, adding other content (such as Central Concept linking) only when necessary. This gives rise to natural, elliptical surface generation. This approach to generation from NewInfo has been developed further by Jokinen et al. (1998) and Jokinen and Wilcock (2003).

26.3.1 Focus Scope in a Semantic Approach

Central Concept (topic) and NewInfo (focus) are represented using QLFs with explicit indices for discourse referents. This facilitates distinguishing old and new information, but the QLF lacks explicit representation of scope. It would be useful to be able to represent focus scope ("narrow focus" and "wide focus"), and also to be able to represent quantifier scope. This issue will be addressed in Section 26.6.

Example 26.6 shows an interesting "disjunctive focus", where the disjunction itself is reified and has its own semantic index. Although many examples of narrow and wide focus can be elegantly represented in the PLUS approach, simply by NewInfo taking the appropriate index value, other examples cannot be represented by a single semantic index: if *hates* has semantic index h, the wide VP focus reading in (26.2) would need NewInfo to be both h and s. It is not possible to unify these indices, because the hating event (h) and the china set (s) are ontologically distinct items. The conclusion is that the value of NewInfo should be a *set* of indices, giving representations like those sketched in (26.7) (narrow NP focus) and (26.8) (wide VP focus):

(26.7) Semantics: hateEvent(h,p,s),president(p),Delft(s),china(s),set(s)
NewInfo: {s}

(26.8) Semantics: hateEvent(h,p,s),president(p),Delft(s),china(s),set(s)
NewInfo: {h,s}

This need for set-valued features, using sets of semantic indices to represent focus scope, is analogous to the need for set-valued features, using sets of syntactic categories, in the approach of Section 26.2.

26.4 Information Structure: A Functional Approach

In Sections 26.2 and 26.3 we described a syntax-oriented approach and a semantics-oriented approach, but our aim is to move towards a discourse-oriented approach to information structure, in which its representation should not be too closely tied to either syntax or semantics. This has long been a fundamental assumption in functionally-oriented frameworks.

For example, Teich (1998) illustrates how focus scope is handled in Systemic Functional Grammar. In the *function structures* in (26.9) and (26.10) there is a syntax-oriented layer (Subject-Finite-Object), a semantics-oriented layer (Actor-Process-Goal), and **two** further layers of discourse-oriented information.

(26.9)

Actor	Process	Goal
Theme	Rheme	
Given	New	
Subject	Finite	Object
Fred	*ate the beans*	

(26.10)

Actor	Process	Goal
Theme	Rheme	
Given		New
Subject	Finite	Object
Fred ate		*the beans*

26.5 Minimal Recursion Semantics

The kind of flat quasi-logical form (QLF) used in PLUS has the disadvantage that it lacks an adequate treatment of quantifier scope. Minimal Recursion Semantics (MRS), developed by Copestake et al. (1997) in the HPSG framework, is a flat indexed quasi-logical form like the one used in PLUS, but MRS provides a solution to the treatment of quantifier scope.

Both MRS and the indexed QLF of PLUS were motivated by the needs of machine translation, where "flat" representations are preferred over strongly head-driven representations, as the head in one language may not correspond to the head in another language. Like the QLF, MRS depends on the use of indices to represent dependencies between the terms in the flat list. Before the development of MRS, HPSG used indices only for entities of type *nominal_-object*, to assign them to semantic roles as participants in *states of affairs* and to carry agreement features. In MRS, indices are also used for events, as in the QLF.

One difference between MRS and the QLF is that MRS uses typed feature structures instead of ordinary logical terms. Each element in the list of semantic terms is an HPSG typed feature structure of type *relation*. This facilitates the integration of MRS into HPSG.

26.5.1 Quantifier Scope in MRS

Another difference, which makes MRS a significant improvement over the QLF, is that MRS supports the representation of quantifier scope, either fully resolved or underspecified. This is done by including *handles* which label each term in the list. (As a musical joke about semantic *composition*, the handle feature is named HANDEL and the list feature is named LISZT by Copestake et al. (1997)).

Scope can be represented by means of the handles, while maintaining the flat list representation, without the nesting required when operators are used to represent scope. The handles are unified with the role arguments of other

relations. This technique not only enables recursive embedding to be simulated, but also allows quantifier scope to be either fully resolved or underspecified. We give an example from Copestake et al. (1997) using their linear notation to save space. The unscoped representation of *every dog chased some cat* is:

(26.11) 1:every(x,3,n), 3:dog(x), 7:cat(y), 5:some(y,7,m), 4:chase(e,x,y)
top handle: p

Here 1, 3, 4, 5, 7 are handles and m, n and p are variables over handles. This unscoped representation can be further instantiated to give scoped representations by unifying m, n and p with the appropriate handles:

(26.12) 1:every(x,3,4), 3:dog(x), 7:cat(y), 5:some(y,7,1), 4:chase(e,x,y)
top handle: 5 (wide scope *some*)

(26.13) 1:every(x,3,5), 3:dog(x), 7:cat(y), 5:some(y,7,4), 4:chase(e,x,y)
top handle: 1 (wide scope *every*)

The top handle allows the clause to be embedded in a longer sentence. In the scoped representations, it is unified with the widest scoped quantifier.

26.6 Information Structure and MRS

If information structure is a distinct dimension, as argued by Engdahl and Vallduví (1996), its representation should not be too closely tied to either syntax or semantics. However, we noted that the semantics-oriented approach had advantages in topic-tracking and distinguishing old and new referents due to its direct use of semantic indices. A representation for use in practical dialogue systems, while not directly tied to either syntax or semantics, should nevertheless be relatively close to the semantic information. We therefore take the MRS representation as a starting point for a representation of information structure in HPSG, but follow Engdahl and Vallduví (1996) in locating INFO-STRUCT in CONTEXT.

To avoid confusion, we also follow Engdahl & Vallduvi's feature terminology: INFO-STRUCT includes FOCUS and GROUND, and GROUND includes LINK and TAIL. However, the values of FOCUS, LINK and TAIL will not be syntactic constituents, they will be variables over handles. These variables will be unified with particular handles in the semantics in order to represent specific focus scopings and topic interpretations. An advantage of handles is that they can be unified with each other without implying that semantic entities lose their distinct identities. This raises the unresolved question whether focus scope can be handled in MRS in a similar way to quantifier scope. However, we will follow the earlier approaches and use set values. In our representation, these will be sets of handles.

We start by adding information structure to the MRS quantifier example of

Copestake et al. (1997), *every dog chased some cat*. If we assume a context (perhaps *what did every dog chase?*) in which *every dog* is interpreted as link, and *some cat* has narrow focus, we can use a representation such as:

(26.14) 1:every(*x*,3,4), 3:dog(*x*), 7:cat(*y*), 5:some(*y*,7,1), 4:chase(*e*,*x*,*y*)
TOP-HANDLE:5, LINK:{1}, TAIL:{4}, FOCUS:{5}

By contrast, if we assume a context (perhaps *what did every dog do?*) in which there is wide focus across *chased some cat*, we need to include handles 4 and 5 in the value of FOCUS, giving:

(26.15) 1:every(*x*,3,5), 3:dog(*x*), 7:cat(*y*), 5:some(*y*,7,4), 4:chase(*e*,*x*,*y*)
TOP-HANDLE:1, LINK:{1}, FOCUS:{4,5}

26.6.1 Focus Scope in MRS

We now sketch new MRS-based representations of some of the examples of Engdahl and Vallduví (1996). The alternative focus scope readings of examples (26.1) and (26.2) can be represented by (26.16) and (26.17):

(26.16) 1:the(*x*,2), 2:president(*x*), 3:the(*y*,4), 4:china(*y*), 4:set(*y*),
5:hate(*e*,*x*,*y*)
TOP-HANDLE:5, LINK:{1}, TAIL:{5}, FOCUS:{3} (narrow focus)

(26.17) 1:the(*x*,2), 2:president(*x*), 3:the(*y*,4), 4:china(*y*), 4:set(*y*),
5:hate(*e*,*x*,*y*)
TOP-HANDLE:5, LINK:{1}, FOCUS:{3,5} (wide focus)

Example (21) of Engdahl and Vallduví (1996), *The president* [F HATES] *the Delft china set*, is straightforward:

(26.18) 1:the(*x*,2), 2:president(*x*), 3:the(*y*,4), 4:china(*y*), 4:set(*y*),
5:hate(*e*,*x*,*y*)
TOP-HANDLE:5, LINK:{1}, TAIL:{3}, FOCUS:{5}

The more problematic subject-verb focus in example (26.3), [F *The* BUTLER BROKE] *the set*, can be represented in MRS by:

(26.19) 1:the(*x*,2), 2:butler(*x*), 3:the(*y*,4), 4:set(*y*), 5:break(*e*,*x*,*y*)
TOP-HANDLE:5, TAIL:{3}, FOCUS:{1,5}

The complex focus in example (26.4) can be represented in MRS as shown in (26.20), using the NAME relation of Copestake et al. (1997).

(26.20) 1:NAME(*x*,John), 2:NAME(*y*,Bill), 3:NAME(*z*,Sue),
5:introduce(*e*,*x*,*y*,*z*)
TOP-HANDLE:5, LINK:{1}, TAIL:{5}, FOCUS:{2,3}

Finally example 26.21 shows one possible MRS-based representation for the PLUS disjunctive focus example in (26.5), *Do you want to* **buy or rent** *one?*.

(26.21) 1:want(w,u,2) 2:or(3,4) 3:buy(b,u,c) 4:rent(r,u,c) 5:car(c), 6:user(u)
TOP-HANDLE:1, LINK:{1}, TAIL:{5}, FOCUS:{2}

26.7 Conclusion

We have compared two different approaches to representing information structure: a syntax-oriented approach proposed in HPSG, and a semantics-oriented approach used in a practical dialogue system. In both cases we noted that the problem of representing focus scope requires the use of set-valued features.

We noted that the Minimal Recursion Semantics (MRS) representation used for HPSG can represent quantifier scope using handles. We proposed in Section 26.6 a way to extend MRS to include information structure. This raises the unresolved question whether focus scope can be handled in MRS in a similar way to quantifier scope. Using a simpler, set-valued approach we showed how narrow focus, wide focus, subject-verb focus, complex focus and disjunctive focus can be treated in this extended MRS representation.

References

Copestake, Ann. 2000. *Implementing Typed Feature Structure Grammars*. Stanford: CSLI Publications.

Copestake, Ann, Dan Flickinger, and Ivan A. Sag. 1997. Minimal Recursion Semantics: An Introduction. Ms. Stanford University.

Engdahl, Elisabet and Enric Vallduví. 1996. Information packaging in HPSG. In C. Grover and E. Vallduví, eds., *Edinburgh Working Papers in Cognitive Science, Vol. 12: Studies in HPSG*, pages 1–32. University of Edinburgh.

Jokinen, Kristiina. 1994. *Response Planning in Information-Seeking Dialogues*. Ph.D. thesis, University of Manchester Institute of Science and Technology.

Jokinen, Kristiina. 2005. Finnish discourse syntax grammar. (this volume).

Jokinen, Kristiina, Hideki Tanaka, and Akio Yokoo. 1998. Planning dialogue contributions with new information. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 158–167. Niagara-on-the-Lake, Ontario.

Jokinen, Kristiina and Graham Wilcock. 2003. Adaptivity and response generation in a spoken dialogue system. In J. van Kuppevelt and R. Smith, eds., *Current and New Directions in Discourse and Dialogue (Text, Speech and Language Technology, Vol. 22)*, pages 213–234. Kluwer Academic Publishers.

Sag, Ivan A. and Thomas Wasow. 1999. *Syntactic Theory: A Formal Introduction*. Stanford: CSLI Publications.

Steedman, Mark. 1991. Structure and intonation. *Language* 67(2):260–296.

Teich, Elke. 1998. Types of syntagmatic grammatical relations and their representation. In *Processing of Dependency-based Grammars: Proceedings of the Workshop, COLING-ACL'98*. Montreal.

Vilkuna, Maria. 1989. *Free Word Order in Finnish: Its Syntax and Discourse Functions*. Helsinki: Finnish Literature Society.

Wilcock, Graham. 1998. Approaches to surface realization with HPSG. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 218–227. Niagara-on-the-Lake, Ontario.

Wilcock, Graham. 1999. Lexicalization of Context. In G. Webelhuth, J.-P. Koenig, and A. Kathol, eds., *Lexical and Constructional Aspects of Linguistic Explanation*, pages 373–387. Stanford: CSLI Publications.

Wilcock, Graham. 2001. Towards a discourse-oriented representation of information structure in HPSG. 13th Nordic Conference on Computational Linguistics, Uppsala, Sweden.

27

Developing a Dialogue System that Interacts with a User in Estonian

HALDUR ÕIM AND MARE KOIT

27.1 Introduction

There are many spoken dialogue applications in different languages available in the world: flight reservation systems worked out in USA within the DARPA programme, flight and train schedule systems developed in Europe within the SUNDIAL programme, the Verbmobil meeting agreement system in Germany, a help desk and bus schedule system developed within the Interact project in Finland, etc (McTear 2004).

No such system is available for Estonian so far. The analysis of actual human-human dialogues is needed in order to find out the variants of universal norms and rules that are typically used in particular language and culture. To get the empirical material, we are collecting Estonian spoken dialogues. The Estonian Dialogue Corpus (EDiC) presently includes about 600 spoken human-human dialogues. A typology of dialogue acts has been worked out and is used for annotating the corpus (Hennoste, Rääbis 2004; Gerassimenko et al. 2004, Hennoste et. al. 2003). The typology is based on the conversation analysis (CA) approach. Dialogue acts are divided into two big groups (1) acts that form adjacency pairs (AP) where the first part requires a certain second part (e.g. questions and answers) and (2) non-AP acts (e.g. acknowledgement).

In this paper, we shall analyse calls for information to find out methods and ways used by people for ordering and giving information, and to model them in a DS. Let us call these methods communicative strategies (cf. Jokinen 1996).

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

27.2 Dialogue System as Conversation Agent

A conversation agent is a program that consists of six (interacting) modules (cf. Koit, Õim 1998):

$$\mathrm{DS} = \{\mathrm{PL}, \mathrm{TS}, \mathrm{DM}, \mathrm{INT}, \mathrm{GEN}, \mathrm{LP}\},$$

where PL – planner, TS – task solver, DM – dialogue manager, INT – interpreter, GEN – generator, LP – linguistic processor. PL directs the work of both DM and TS, whereby DM controls the communication process and TS solves domain-related tasks. The task of INT is to make the semantic analysis of a partner's utterances and that of GEN is to generate semantic representations of agent's own contributions. LP carries out linguistic analysis and generation. The conversation agent uses a knowledge base KB in its work. In our model, the KB consists of four components: $\mathrm{KB} = (\mathrm{KB_W}, \mathrm{KB_L}, \mathrm{KB_D}, \mathrm{KB_A})$, where $\mathrm{KB_W}$ contains world knowledge, $\mathrm{KB_L}$ linguistic knowledge, $\mathrm{KB_D}$ knowledge about dialogue and $\mathrm{KB_A}$ knowledge about interacting agents. $\mathrm{KB_A}$ has two parts: the knowledge of DS about itself and a partner model – the knowledge about a 'standard' user. A necessary precondition of for communication is existence of shared knowledge of interacting agents (e.g. a common picture of the world, a common language of interaction).

27.2.1 Frames of Dialogue Acts

The DS must be able to recognize a user's acts and generate its own responding acts. The full processing cycle of a dialogue act pair can be represented as follows:

speech recognition ⇒ text analysis ⇒ task solving ⇒
text generation ⇒ speech synthesis

The communicating agents exchange acts that express their goals. DS as a cooperative partner must take over the user's goal (in our case, the goal is to get information) and try to fulfil it. Therefore, the dialogue knowledge $\mathrm{KB_D}$ of a DS must include descriptions of dialogue acts that make it possible to infer user's goals. Dialogue acts can be represented as frames having the slots SETTING, GOAL, PLOT and CONSEQUENCE (cf. Saluveer, Õim 1985). The slot SETTING gives preconditions of the dialogue act, including the author's beliefs about the addressee (as a part of the partner model) that can be true or false. An unexpected reaction by the partner signals that a belief was wrong. For example, a user asks for a bus timetable supposing that the DS knows it but actually the data base includes only flight information. If the SETTING is not satisfied then the speaker initiates a subdialogue – asks a specifying question or initiates a repair.

Let us consider the frame of 'closed yes/no question'[1] (the idea is taken from Saluveer, Õim 1985; cf. Bunt 1999; Jurafsky, Martin 2000). The following notations are used: S speaker (author of the act), H hearer (addressee), p proposition (true or false, e.g. *This is a direct bus*). Both the user and the DS perform the roles of S and H alternately.

QUF:CLOSED_YES/NO
SETTING:
S has a wish to know whether p (or not-p)
S believes that H knows whether p
GOAL: H knows that S has a wish to know whether p
PLOT: S informs H that S has a wish to know whether p
CONSEQUENCE: H knows that S has a wish to know whether p

Example: *Is this a direct bus?*

27.2.2 User Model

For a dialogue system, a user (client, C) is a conversation agent like itself. In its work, DS supposes that C has analogous six processing modules and four knowledge bases as the DS itself, and that the intersection of its knowledge bases with those of the user is not empty (otherwise, the interaction would be impossible).

Let us consider the KB_A component of the knowledge base. It includes the knowledge of DS (1) about itself, and (2) about a 'standard' client – his/her beliefs, desires, intentions, and algorithms that are used to generate plans. In the case of information dialogues, a client's beliefs, desires and intentions are related to ordering and getting some information. When asking a question, C believes that DS has the needed information, and his/her intention is to get this information. When analysing a question, DS recognises C's beliefs and intention, and tries to satisfy his/her goal, i.e. to provide him/her the asked information. Therefore, a BDI model which operates with agent's beliefs, desires and intentions can be implemented here (cf. Allen 1995; Koit, Õim 1998; Koit, Õim 2004).

Every question or directive sets up a (new) goal that is reached if a requested answer is received. If C's goals are unsatisfied after (s)he got an answer, then (s)he initiates a clarification subdialogue, asking a new, specifying question. The dialogue manager must keep accounts of C's beliefs and goals. A suitable data structure is a stack. Every question/directive (the first part of

[1] Frame names (equal to dialogue act tokens in our typology of dialogue acts) are originally in Estonian. Every token consists of two parts separated by a colon: the first two letters form an abbreviation of the act group name (e.g. QU = question). The third letter is only used for AP acts: the first (F) or the second (S) part of an AP act. The second part of a token is the full name of the act.

an AP) adds a new goal (subgoal) into the stack, and every answer or fulfilling a directive (the second part of the AP) may delete the upper goal.

27.3 Information-Sharing Strategies in Estonian Spoken Dialogues

27.3.1 Overview of Empirical Material

20 institutional dialogues (calls for information) were chosen from EDiC where a client (C) orders information and an information provider (P) provides them. The calls are short – the average length of a dialogue is 13 utterances. The total number of utterances is 275 and the number of words – about 1,000. A typical call consists of three parts: a conventional beginning, main information part, and a conventional ending. The kernel of the information part is a question – answer (or directive – grant) AP: a question is asked (or a request made) and an answer (or grant) is obtained. Still, subdialogues can occur after a question and/or answer: an adjusting/specifying question is asked and answered, or a repair for solving a communication problem is initiated and performed. The kernel can be repeated; more than one question can be asked and answered. In the analysed dialogues, C asks for a phone number in most cases (Table 1).

Table 1. What is being asked for

Client's goal	Number of dialogues
Phone number	16
Bus time	2
Film in cinema	1
Start of street	1
Address	1

Typically, C has only one goal (17 dialogues from 20), e.g. to obtain a phone number. The goal is reached after the answer is received, and then the conversation can be finished.

In the remaining 3 dialogues, C has more than one goal. In the first of them, C asks for the phone number of one travel bureau and then of another. These two goals can be considered as subgoals of a general goal – to take a trip (but it is beyond this dialogue). In the second dialogue, C similarly asks two questions, both of which are about a bus departure time. Supposedly, C intends to take a bus. In the third dialogue, C's first four questions (which film, the start time today evening and tomorrow morning, the price of a ticket) point to his/her intention to go to the cinema. The last question concerns a phone number and is not connected with going to the cinema.

27.3.2 Dialogue Acts that Set Up Goals

All of the analysed dialogues have a standard beginning part – P responds the call by saying the name of the company (e.g. *Estmar information*[2]), introducing himself/herself (e.g. *Leenu is hearing*) and greeting (*good morning*). Typically, C responds to the greeting and immediately requests information. Every question or request sets up a goal. In a cooperative conversation, P will share C's goal and assist C in reaching it.

A limited number of dialogue acts are used to express the goal (Table 2): request (e.g. *give me the teachers room of the Karlova school*), an indirect speech act which we call open yes/no question (*could you tell me the departure time of the bus to Tallinn*), or alternative question (*where does the Aleksandri Street begin—at the town centre or at the other end*). An advance note sometimes precedes a request (*I have a question*), or an additional information follows (*[the ticket office of the theatre Vanemuine,] such a place where tickets can be bought, please*). Some new goals are set up in such cases where P is not able to fulfill the request and offers substituting information (e.g. a phone number where C can get information).

In many dialogues, C starts his/her request with a cue phrase which precisely determines the following dialogue act: a request (*I wanted to know*), open yes/no question (*could you tell me*). Such phrases provide good features for automatic recognition of the dialogue act type (pragmatic analysis) and its meaning (semantic analysis).

P does not always succeed in giving a sufficient answer in the analysed dialogues. C obtains the requested information only in half of the cases, and substituting information in seven dialogues. (S)he does not get any information in three dialogues (e.g. the requested phone number is missing in the data base), therefore his/her goal will not be reached.

Table 2. Dialogue acts used by clients

[2] The examples are translated from Estonian.

Client's dialogue act	Typical phrases	Number of cases
request	*I'd like to know, please*	13
open yes/no question	*could you tell, is it possible to know*	6
accept of an offer/request	-	4
advance note + open yes/no/ alternative question	*I have such a question, one more question*	2
wh-question	*please tell me*	1
request + additional information	-	1

27.3.3 Communicative Strategies Used by Client

Calls for information form a simple dialogue type where the client has only one certain question in most cases. We found 27 questions/requests of C in analysed 20 dialogues.

P recognized C's goal immediately in 16 cases (Table 3) and either provided the requested information or informed C that it was missing. The information part of a dialogue consists of one adjacency pair of dialogue acts. In the remaining cases, C did not formulate his/her question precisely enough, and a subdialogue was started in the ensuing process. The initiator of a subdialogue is either P or C. In one dialogue, C reformulated his request three times, and in another dialogue, specified the answer (C: *Is it near the department store?* P: *Farther away, to the Lille hill.* C: *Is Lille the street which goes from the department store?*).

Table 3. How a client orders information

Strategy	Number of requests/questions
Request/question that does not need adjusting	16
Request/question that needs adjusting	
(a) client initiates adjusting	5
(b) information provider initiates adjusting	4
Request/question that needs reformulation	2

27.3.4 Communicative Strategies Used by the Information Provider

In a typical case, P gives the asked information either immediately or after adjusting (in 20 cases out of 27). If the requested information is missing then P either offers substituting information, or behaving non-cooperatively, does not offer anything (Table 4).

Table 4. How an information provider gives information

Strategy	Number of answers
The needed information exists and is provided immediately	12
The needed information exists and is provided after adjusting initiated by the client	5
The needed information exists and is provided after adjusting initiated by the information provider	4
The needed information does not exist; the provider offers a substitution	4
The needed information does not exist; the provider does not offer a substitution	2

In seven dialogues out of 20, P initiates an information-sharing subdialogue before answering. The subdialogue always consists of one AP of dialogue acts (cf. Hennoste et al. 2005): P's alternative question or wh-question followed by C's giving information, or P's question which offers an answer (sometimes clarification) followed by C's agreement. An information-sharing subdialogue explains which information C needs, and helps him/her to reach the goal. C reaches the original goal in three cases and gets substituting information in three cases. The answer turns out to be wrong in one case (the phone number in another town).

Information providers are specially trained to tell phone numbers. In the analysed calls, phone numbers consist of three, five or six digits. In case of three digits, all the digits are spelled out in sequence which the client acknowledges (*mhmh*). A number of five digits is spelled out in two parts – two and three digits separated by a micropause. C either repeats all the digits, or the last three ones, and P confirms (*yes*). A number of six digits similarly is given in two parts – the first three and the last three digits. C's response de-

pends on the length of pause between the two parts. In case of a long pause, C either repeats all the first three digits, or acknowledges them (*yes*). C always repeats all the last three digits. Sometimes (s)he adds the word *yes?* waiting for P's confirmation.

27.4 Information Provider as a Conversation Agent

The DS which performs the role of information provider implements a formal grammar for dialogue management (Figure 1). The grammar is based on APs of dialogue acts. When requesting information, a client uses the first part of an AP: question (QUF) or directive (DIF). Dialogue act names are terminals of the grammar (capital letters are used in act names).

The DS uses a stack to keep shared goals. C's request or question sets up a goal which goes to the stack. If DS needs additional information for answering then it initiates an information-sharing subdialogue by asking an adjusting question. The question asked sets up a subgoal of the original goal and goes to the stack onto the original goal. When the answer is obtained then the goal will be removed from the stack. If the stack is empty then all the goals have been achieved (Table 5).

Table 5. Example of using a goal stack

Utterance	**Dialogue act**	**Goal stack**
/---/		
C: *tell me please the phone number of the dentist Vigoroovit*	DIF:REQUEST	
P: *where the dentist is located*	QUF:WH-QUESTION	Phone number
C: *2, Tuglase Street*	QUS:GIVING_INFORMATION	Address Phone number
P:		~~Address~~ Phone number
/---/		

The described ideas and results of corpus analysis are implemented only partly at the moment. A DS is being worked out (author Margus Treumuth)

Notation	Number of z's
$(z)^+$	one or more
[z]	zero or one
$(z)^*$	zero or more

information_dialogue ::= beginning main_part ending
beginning ::= [RIF:INTRODUCTION] RIF:GREETING
RIS:GREETING
ending ::= RIF:THANKING RIS:PLEASE [RIF:GOODBYE
RIS:GOODBYE]
main_part ::= (ordering_information (ordering_information
giving_information)* (giving_information)$^+$
(ordering_information giving_information)*)$^+$
ordering_information ::= Questions_first | Directives_first |
(advance_note)* ordering_information
giving_information ::= Questions_second | Directives_second |
giving_information (additional_information)*
advance_note ::= SA:ADVANCE_NOTE
additional_information ::= AI:SPECIFICATION | AI:ASSESSMENT
Questions_first ::= QUF:CLOSED_YES/NO | QUF:OPEN_YES/NO |
QUF: ALTERNATIVE | QUF:WH-QUESTION |
QUF:OFFERING_ANSWER
Directives_first ::= DIF:REQUEST | DIF: PROPOSAL | DIF:OFFER
Questions_second ::= QUS:YES | QUS:NO | QUS:AGREEING_NO |
QUS:ALTERNATIVE:ONE | QUS:ALTERNATIVE:BOTH |
QUS:ALTERNATIVE:THIRD_CHOICE |
QUS:ALTERNATIVE:NEGATIVE |
QUS:GIVING_INFORMATION |
QUS:MISSING_INFORMATION | QUS:REFUSAL |
QUS:POSTPONEMENT
Directives_second ::= DIS:GIVING_INFORMATION |
DIS:MISSING_INFORMATION | DIS:REFUSAL |
DIS:AGREEING | DIS:DISAGREEING |
DIS:RESTRICTED_AGREEING | DIS:POSTPONEMENT

Figure 1. Grammar of a simple information dialogue

which interacts with the user in Estonian and gives information about the flights departing from the Tallinn airport. The user inserts his/her question on a web page (http://www.ut.ee/~treumuth/) in the form of a written sentence or phrase in Estonian, and gets an answer in form of text and/or synthesized speech. The world knowledge base KB_W contains information of flight times and destinations. The linguistic processor LP performs a morphological analysis of the user's utterances in order to find out the cue words, and uses ready-made sentence templates with some word forms generated by the morphological synthesis to compile the answers. The text-to-speech module is integrated into the DS.

27.5 Conclusion

We have analysed spoken human-human dialogues in Estonian with the aim of investigating how people request and receive information. Some information-sharing strategies used by clients and information providers have been established. DS that performs the role of an information provider is a conversation agent which consists of various functional blocks and uses various knowledge bases in its work. The dialogue management block uses a formal grammar of dialogue acts. The grammar expresses the idea of adjacency pairs of dialogue acts – one of fundamental ideas of conversation analysis. Every question or request of a client (the first part of an AP) is expecting an answer (the second part of the corresponding AP). Every question and request sets up a new goal or subgoal. DS as a cooperative partner shares client's goals. A stack is used for these shared goals. Every satisfactory answer removes a goal from the stack. For the DS, a user is a conversation agent similar to itself. Beliefs, desires and intentions of a user must be taken in account in order to give him/her the needed information. This work is still in progress. Our further work will concentrate on finding out of more detailed communicative strategies and on formal definitions of more dialogue acts that make it possible the automatic recognition of user goals in a cooperative dialogue system.

Acknowledgement

This work is supported by Estonian Science Foundation (grants 5685, 5534).

References

Allen, J. 1995. *Natural Language Understanding*. 2nd ed. The Benjamin/Cummings Publ. Comp., Inc.

Bunt, H. 1999. Dynamic Interpretation and Dialogue Theory. *The Structure of Multimodal Dialogue I.*, ed. M.M. Taylor, F. Neel, and D.G. Bouwhuis, 139–166. Philadelphia/Amsterdam: John Benjamins.

Hennoste, T., O. Gerassimenko, R. Kasterpalu, M. Koit, A. Rääbis, K. Strandson, M. Valdisoo. 2005. Information-Sharing and Correction in Estonian Information Dialogues: Corpus Analysis. *Proc. of the Second Baltic Conference on Human Language Technologies*. 249–254. Tallinn.

Gerassimenko, O., T. Hennoste, M. Koit, A. Rääbis, K. Strandson, M. Valdisoo, E. Vutt. 2004. Annotated Dialogue Corpus as a Language Resource: An Experience of Building the Estonian Dialogue Corpus. *Proc. of the First Baltic conference "Human Language Technologies. The Baltic perspective"*, 150–155. Riga.

Hennoste, T., M. Koit, A. Rääbis, K. Strandson, M. Valdisoo, E. Vutt. 2003. Directives in Estonian Information Dialogues. *Text, Speech and Dialogue. 6th International Conference TSD 2003*, ed. V. Matousek, P. Mautner, 406–411. Springer.

Hennoste, T., A. Rääbis. 2004. *Dialoogiaktid eesti infodialoogides: tipoloogia ja analüüs*. Tartu: TÜ.

Jokinen, K. 1996. Cooperative Response Planning in CDM: Reasoning about Communicative Strategies. *TWLT11. Dialogue Management in Natural Language Systems*, ed. S. LuperFoy, A. Nijholt, G. Veldhuijzen van Zanten, 159–168. Enschede: Universiteit Twente.

Koit, M., H. Õim. 2004. Argumentation in the Agreement Negotiation Process: A Model that Involves Natural Reasoning. *Proc. of the Workshop W12 on Computational Models of Natural Argument. 16th European Conference on Artificial Intelligence*, 53–56. Valencia, Spain.

Koit, M., H. Õim. 1998. Developing a Model of Dialog Strategy. *Text, Speech and Dialogue. International Conference TSD 1998*, 387–390. Springer.

Saluveer, M., H. Õim. 1985. Frames in linguistic descriptions. *Quaderni di Semantica* 6(2):282–292.

McTear, M.F. 2004. *Spoken Dialogue Technology. Toward the Conversational User Interface*. Springer.

28

The Story of Supposed Hebrew-Finnish Affinity - a Chapter in the History of Comparative Linguistics

TAPANI HARVIAINEN

Enevaldus Svenonius was born in the parish of Annerstad in Småland, Sweden, in 1617. He studied at the Universities of Turku (Academia Aboensis in Turku, Finland)[1] and Uppsala; the degree of *magister* was conferred on him by the Faculty of Philosophy in Turku in 1647. Svenonius continued his studies in Uppsala and Wittenberg and travelled widely in Bohemia, Austria, Hungary, Bavaria, Alsace, Switzerland, and the Netherlands in 1654. In the same year he was chosen as *Professor eloquentiæ* (i.e. Professor of Latin) at the Academia Aboensis and six years later, in 1660, he was appointed *Professor Theologiæ* at the same University. Finally, in 1687, the King of Sweden nominated Svenonius as Bishop of Lund and Vice Chancellor of the University in the same city. However, in spring 1688 Svenius died in Turku where he was buried in the Cathedral.

Svenonius was the most productive writer and the leading person in cultural, academic, and church life in Finland in the seventeenth century. Among his extensive literary output *Tò nóēma ēkhmalōtisménon seu potius Gymnasium capiendae rationis humanae,* an encyclopaedic collection of twenty dissertations published in the Faculty of Philosophy in 1658-1662, is the most central work to be dealt with in this context.[2]

I. A. Heikel, who wrote his still indispensable *Filologins studium*

[1] The city of Turku is called Aboa in Latin and Åbo in Swedish.

[2] Seppo J. Salminen has written an extensive scholarly biography of Svenonius: *Enevaldus Svenonius* 1 & 2 (Suomen Kirkkohistoriallisen Seuran toimituksia 106 & 134, Helsinki-Rauma 1978 & Helsinki-Jyväskylä 1985).

vid Åbo universitet ('The study of philology at the University of Åbo') in 1884, includes the following statement in his presentation of Svenonius (p. 57): "As far as is known, even the questionable merit of being the first to propose the sentence that to the greatest extent the Finnish language has received its vocabulary from Greek and Latin, rests with Svenonius."[3] As a rule, a similar amused tone accompanies the descriptions of the linguistic achievements of Svenonius and his colleagues of the seventeenth and eighteenth centuries in both scholarly and popular works, inclusive of textbooks.[4]

Indeed, Svenonius wrote in *Tò nóēma ēkhmalōtisménon seu potius Gymnasium capiendae rationis humanae* (Book 5, Para. XLIII, p. 87) that "Finnicæ lingvæ originem quod concernit, videtur ea maximam esse partem ex Græcis & Hebræis generata vocabulis" ('concerning the origins of the Finnish language, it seems to originate to the greatest part from Greek and Hebrew words'). As examples to prove his statement, he first refers to thirteen Greek words and proper names with their supposed counterparts in Finnish: Greek *khaláō,* Lat. demergo, 'to sink, submerge' = Finnish Kala, Lat. piscis, 'a fish'; Greek *kheīlos,* Lat. labium, 'a lip' = Finnish kieli Lat. lingva 'a tongue'; Greek *khōiros,* Lat. porcus, sus, 'a pork, pig' = Finnish koira, Lat. canis, 'a dog'; Greek *aigésippos,* 'Hegesip' = Finnish Sippi; Greek *basilios* = Rus[sian, sic !] Wasiliwitz &c.

In contrast to Greek,[5] "the Hebrew vocabulary of Finnish" presented

[3] "Svenonius tillkommer äfven den tvifelaktiga förtjänsten att, så vidt man vet, först ha uppstält den satsen, att finskan till största delen har sina ord från grekiskan och hebreiskan", I.A. Heikel, *Filologins studium vid Åbo universitet* (Åbo universitets lärdomshistoria, 5. Filologin. Skrifter utgifna av Svenska Literatursällskapet i Finland, XXVI. Helsingfors 1894, p. 57); Svenonius and his linguistic views are described by Heikel on pp. 51-62, while later proponents of the Hebrew background of the Finnish language are introduced on pp. 149-151 and 208-212.

[4] See e.g. Salminen's summary of the philological parts of Svenonius' work and his sources: Baazius, Scaliger, Beckmann, Glandorp, Walther, Walper, etc.; for the discussion of the Hebrew-Finnish relations Salminen has been unable to find earlier sources (Salminen 1978: 238-260).

[5] Still in 1774 Nils Idman defended the community (gemenskap) of the Greek and Finnish languages with a reference to hundreds of similar words in his extensive work *Försök at wisa gemenskap emellan finska och grekiska språken, såsom tjenande till uplysning i finska folkets historie* written in Swedish (Åbo 1774, 92 pp.) which in 1778 also appeared in French translation in Strasbourg *(Recherches sur l'ancien peuple finois, d'après les rapports de la langue finoise avec la langue grecque*, par M. le pasteur Nils Idman, ouvrage traduit du suédois par M. Genet le fils, Strasbourg: Bauer et Treuttel, 1778, xvi+149 pp.).

Similarly, still in 1770 Nicolaus Funck defended the close relation of Swedish to Greek in his dissertation *De harmonia linguæ Græcæ & Sviogothicæ* at the University of

by Svenonius, one of the first scholars of the local language of his university town, has - to the best of my knowledge - never been published in a form comprehensible to a modern-day student of the history of linguistics whose knowledge of Hebrew and/or Finnish may often be rather limited.[6] Thus the following decipherment may not be out of place in this collection; at the same time it endeavours to provide the reader with an opportunity to realize with the development that took place in the study of Hebrew-Finnish relations during the following century.

Svenonius presents the 36 or 37 Hebrew words in a type of transcript, and their Finnish counterparts are not always easy to identify. In the list below I first give the genuine Hebrew spelling followed by the transcript of Svenonius and then a transcript in a more systematic form based on the academic pronunciation tradition current in those days (N.B.: ch = [x], z = [z], and ts = the affricate [c]). The translations of the Hebrew words into Latin provided by Svenonius are translated by me into English between brackets; after an equation sign it is followed by the Finnish counterpart of the Hebrew word according to Svenonius (underlined by me and a few times clarified with modern spelling / form between brackets). The translations of the Finnish words by Svenonius into Latin (and a few times into Swedish) and their renderings from Latin into English, added by me between brackets,[7] complete the entries. A similar method of presentation is also applied in other vocabularies in this article.

אבה Avah; ava; voluit (‘he wanted, wished’) = <u>åwi</u> (= ovi) (in Swedish) döör (‘a door’) / q: ad nutum patens (‘opening according to wish’).

אוי Oi; oy; Wæ (‘oh’) = <u>woi</u> part. intendendi (exclamatory particle).

אודות Odot; odot; causæ (‘on account of’) = <u>ådotta</u> expectare & q: causas rimari (‘to wait, expect’ & ‘to search for reasons’).

אם Em; em; mater (‘a mother’) = <u>Ämi</u> (= ämmä) anus (‘an old woman’).

איל Ajal, ail; ayal, ayil; ceruus, dux (‘a deer’, ‘a leader’), אילות Ejaluth; eyalut; fortitudo (‘power’) = <u>jalo</u> præstans (‘excellent’).

אכן Achen; achen; verè, profectò (‘surely’) = <u>niniken</u> (= niin ikään) ita, <u>propemodum (‘thus’, ‘similarly’).</u>

Uppsala; parallel ideas concerning the relation between German and Greek and French and Greek were proposed by well-known scholars till the end of the eighteenth century.

[6] Both Heikel (1894: 56-57) and Salminen (1978: 240-241) quote a number of Greek etymologies of Finnish words in Svenonius; however, the similar lists of the Hebrew vocabulary have remained beyond their scope.

[7] In a number of cases Svenonius’ Latin equivalents of Finnish words are inaccurate; however, in this context these errors are irrelevant and are not corrected by me.

הולל Holel; holel / holal; vesanus ('furious, madman') = hullu insanus ('folly, infatuated').

חזה Chadsah; chaza; vidit ('he saw') = katzo idem.

חטב Chatab; chatav; cæcidit ligna ('he cut firewood') = catawa (= kataja) juniperus ('a juniper').

חלף Chalaph; chalaf; penetravit ('he passed on, penetrated') = kelpa juvare q: opem insinuare ('to help something to insinuate').

חמל Chamal; chamal; clemens f. ('he had compassion') = camala mirabilis : clementia enim Dei quòd milliès superet justitiam mirari subit ('surprising, awful : namely, the compassion of God which a thousand times exceeds the justice is surprising').

חמר Chamar; chamar; lutosus f. ('was muddy') = camara pellis suilla, q: semper lutosa ('pigskin which is always muddy').

חרמש Chærmæsch; chermesh; falx messoria ('a harvest sickle') = kermess (= kärmes, käärme) serpens, à simili figurâ ('a serpent, from a similar shape').

חרפה Cherpah; cherpa; probrum ('shame'), cui non dissimilitèr enunciatur membrum virile ('with which not dissimilarly the male organ is called').[8]

ילך Ialach; yalach; ivit ('he walked', a theoretical verb which in practice does not occur in Hebrew) = jalka pes ('a foot, leg').

יפיח Iapheach; yafeach; efflavit, locutus est ('it blew', 'he spoke'), & פוח poach; poach / puach; flare ('to breathe') = poho, (Swedish) bläsa ('to blow') / puhu, (Swedish) tala ('to speak').

עמנו אל Imanuel; imanu'el; *anagrammatistheīs* ('God is with us' with letters in a different order) = Jumalen : Jumala enim, quod Deum significat ('God's : God, which signifies God').[9] Svenonius continued by writing that it is rather probable that Jumala should be derived from Hebrew יום Iom; yom; dies ('a day'), & מלא Mala; mala / male; plenus f. ('was full'), q: plenus dierum & annorum, ut significet idem quod infinitus & æternus ('i.e. full with days and years to signify Him who is infinite and eternal').[10]

[8] With a tacit reference to the Finnish word kyrpä 'penis', not in polite use.

[9] After this equation Svenonius adds that this etymology is preferable to that from Julma ('terrible'), which more probably is derived from Jumala ('God').

[10] Svenonius goes on to argue that in Finnish the letter o is easily pronounced as [u]; the latter etymology accords well with the Scriptures, because God the Father is called the Ancient of Days (Dan. 7,13. 22), and God the Son proceeds from ancient days (Micah 5,2), whose years will never end (Hebrews 2,12 [an error *pro* Heb. 1,12]). In plural *Iom* (day) refers

כנס Canas; kanas; collegit (‘he collected’) = Kansa (= kanssa) cum (‘with’).
כר Car; kar, Camelus, agnus, aries (‘a camel’, ‘lamb’, ‘ram’) = Karia, S[wedish] boskap, pecudes (‘cattle’).
ליש Laisch; layish, Leo decrepitus (‘a decrepit lion’) = Laiska piger (‘lazy’).
נשל Naschal; nashal; solvit (‘he loosened, undid’) = Nascala subula (‘a cobbler’s awl’).
סוס Sws; sus; 1. Equus 2. Grus, 3. Anser sylvestris, variorumquè aliorum animalium nomen (‘1. a horse, 2. a crane, 3. a wild goose, and the name of various other animals’) = Susi Lupus (‘a wolf’).
סלה Silla; (theoretically) sil-la; stravit (‘he built a way’) = Silla (= silta) pons (‘a bridge’).
סלח Sallach; sallach; condonavit (‘he forgave’)[11] = salli permittere (‘to permit’).
עלפה Ulpæ; ulpe;[12] obtectus ore (‘with a mouth covered up’) = ylpiä superbus (‘proud’).
פורה Purah; pura; in quod uvæ confringendæ mittuntur (‘in which the grapes to be pressed are put’) = Puro puls (‘a brook’).
פימה Pimah; pima; omentum, pingvedo (‘the fatty membrane or caul covering the intestines’, ‘fatness’) = Pimä pingvedo lactis (‘butterfat, buttermilk’).
פסה Pissah; pissa; particula (‘a particle’)[13] = pissar (= pisara) guttula (‘a small drop’).
פרש pæræsch; peresh; fimus æquiv. met. podex (‘manure, metonymically equal to the anal orifice’).[14]
צרע Tsara; tsara; leprosus f. (‘was leper’) = sairas ægrotus (‘ill’).
קדח Kadach; kadach; accendit, ferbuit (‘was kindled’, ‘glowed’) = Kådas (= kota) culina (‘cooking hut’).
קדים Kadim; kadim; ante pridem[15] (‘in front, before’ and ‘in days of yore’) = kodast (= kohdast = kohdakkain) è regione (‘opposite’).
קול Kool; kol; sonus (‘a voice, sound’) = kuula audire (‘to hear’).
ראה Raah; ra’a; vidit, providit, pavit (‘he saw’, ‘predicted’, ‘provided’, ‘was

to years; God the Holy Spirit proceeds from both eternities (i.e. the past and the future, TH) and he is the spirit of the veritable eternity (John 15,26; Ps. 33,6 & 119,90).

[11] More correctly ‘ready to forgive’, occurs only in Ps. 86,5.

[12] A corrupt word in Ezek. 31, 15.

[13] An unexplained word occurring in Ps. 72,16.

[14] With a tacit reference to the Finnish word perse ‘buttocks’, not in polite use.

[15] Obviously meant to have a comma after *ante*.

afraid') = Raha pecunia, quà sibi quis providet de victu & amictu ('money which everyone provides for himself concerning food and clothing').

רוח Roach; roach / ruach; spirare ('to blow, breathe') = Roka (= ruoka) cibus, quo spiritus sive vita sustentatur ('food by which the spirit or life is sustained').

רפה Rippah; rippa; debilitavit ('he weakened') = Råpi (= rupi) assumentum caducum ('a disappearing patch' = 'scab'). &c.

From the viewpoint of later centuries the equations of Svenonius look more or less casual and even ridiculous, as has been stated in numerous contexts.

In 1692 Eric Wallenius defended the dissertation *De confusione lingvarum*[16] under the *præsidium* of Daniel Johannis Lund, Professor of Oriental languages and Greek at Academia Aboensis; in this work the Finnish language was concluded to possess "not only minor vestiges" of the languages which were spoken before "the confusion of languages"; these are to be found in the vocabulary and affixes in particular.[17] A more detailed discussion of the similarities was not included in the booklet, however.

Five years later, on November 13, 1697, the theme of the equivalence of Hebrew and Finnish was dealt with, again under the *presidium* of David Lund, in the *pro gradu* (magister) dissertation *Lingvarum ebrææ et finnicæ convenientia* presented by Eric Erici Cajanus (1675-1737) at the same University in Turku.[18]

At first, Cajanus was able to find equivalent words in Hebrew and Finnish; due to the limited space in his dissertation – he wrote – he enumerated (p. 8) only six words (four of them occurred in Svenonius!) "although a

[16] [Aboæ 1692, 22 p.], Jorma Vallinkoski, *Turun Akatemian väitöskirjat 1642-1828 - Die Dissertationen der alten Universität Turku (Academia Aboënsis) 1642-1828* (Helsingin yliopiston kirjaston julkaisuja - Publications of the University Library at Helsinki 30, Helsinki 1962-1969 = Valllinkoski), No. 2325; *Suomen kansallisbibliografia - Finlands nationalbibliografi - Finnische Nationalbibliographie,* I-II (ed. Tuija Laine & Rita Nyqvist, Vammala-Helsinki 1996 = SKB), No. 2448; Heikel 1894:149-150.

[17] "Cum hanc linguarum examina confusionem, unicum hoc tantum bonâ veniâ paceque eruditorum, expers tamen affectatæ laudis dixerim, scilicet idioma Finnonicum haud exigua primævi præ se ferre vestigia, quod ut existimem, tum plurimarum vocum affinitas, tum affixorum similis indoles mihi persvadet" (p. 14).

[18] [Aboæ 1697, 16 pp.], Vallinkoski No. 2350; SKB No. 2476.

Daniel Lund was born in Halikko in southern Finnish-speaking Finland and Cajanus in Sotkamo, in northern Finnish-speaking Finland; thus, in contrast to the Swedish Svenonius, they knew Finnish well.

more extensive list easily could be collected":[19]

אם em, mater ('a mother') = emä ('a mother').
זה ze, pron. Demonstrativum, iste ('this') = Se ('this', 'it').
הביש hevish, hiph. â בוש pudefacere ('to make ashamed') = häväistä ('to make ashamed').
הולל holel / holal, insanus ('folly, infatuated') = hullu ('folly, infatuated').
חזה chaza, vidit ('he saw') = catzo ('has watched').
ילך yalakh, ivit (theoretically 'he walked') = jalka pes ('a foot', 'a leg').

However, Cajanus was not satisfied with a word list. According to the traditions of the linguistic studies of those days, he continued to examine the various parts of speech (partes orationis) of Hebrew and Finnish – though he does not mention this self-evident attitude in his work. Cajanus was able to make the following observations: In the morphology Finnish reveals counterparts to three out of the four "conjugations" (i.e. stems) of Hebrew verbs *(Kal* teki fecit 'he made', *Pihel* teeskeli factitavit 'he frequented/used to make', and *Hiphil* teetti facere permisit 'he let make'). Both languages possess independent and non-independent forms of personal pronouns; among the independent pronouns the plural forms of Hebrew *'attem* 'you' and *hem* 'they' closely resemble their Finnish counterparts te and he, while the non-independent short forms can be added as (possessive) suffixes to a noun (e.g. Hebr. *sifre**nu*** kiria**mme** libri nostri 'our books', cf. Hebr. *'anaḥ**nu*** and Finnish me 'we'). Further, in both languages these pronominal suffixes can be attached to verbs (i.e. infinitives); thus e.g. *'okhli, 'okhlekha,* and *'okhlo,* derivations of the verb *'akhal* 'to eat', meaning *edere me/te/eum,* correspond to the Finnish expressions syödesäni, syodesäs, and syödesäns ['when I/you/he eat(s)];[20] these forms also imply transformations of the vowel patterns in the two languages. In poetry the metre which usually consists of eight syllables as well as the recurrent parallelism of two verses are no minor proofs of the affinity. In the syntax it is worth noticing that for the address both languages apply the second

[19] The transcriptions and English translations have been added by me.

[20] Still in 1858 these Finnish suffixes were mentioned by G.L. Pesonius as an exceptional feature shared by "other Semitic languages, too" [p. 287: "Vielä sitte on suomella, niinkuin muillakin Semitan kielillä, liitettäviä asemoita (latinaksi pronomina sufucsiva), jotka muilta kieliltä tykkänään puuttuu."]; Pesonius was the first Rector of the first Finnish *gymnasium* in Jyväskylä who also served as the Lecturer in Religion, Greek, and Hebrew in the same school. Gottlieb Leopold Pesonius, 'Rehtorin puhe Jyväskylän ylä-alkeiskoulun avajaisissa 1. 10. 1858', published e.g. in: *Suomen sana* (Suunn. ja toim. Yrjö A. Jäntti, Porvoo 1965): 285-288.

person singular; instead of the various degrees of comparison of adjectives a reduplicated positive form or a positive form added with an emphatic word (Hebrew *me'od*, Finnish *aiwan,* 'very') replace superlatives in both Hebrew and Finnish. Two consonants in initial position cannot occur in these languages.

A comparison between the arguments of Svenonius and Cajanus is interesting. Svenonius introduced the presumption of the equivalence of Hebrew and Finnish. However, as evidence in favour of his statement he was able to propose a mere list of similar words – the unsteady similarity of which probably casted suspicion on the theory even in his time; on the basis of very similar lists Svenonius also defended special contacts of Swedish with Latin, Greek and Hebrew, on the one hand, and of Latin with Greek and Hebrew, on the other.[21] Nevertheless, in his time Svenonius was an authoritative scholar whose conclusions constituted a starting-point for further research.

Instead of a list of words Eric Cajanus penetrated the question on a more comprehensive level: he examined all the parts of speech which, according to the grammarian tradition of his period, were considered to characterize the very essence of a language.[22] In addition to a condensed list of lexical similarities Cajanus was able to point out similarities in the morphology, prosody, syntax, and phonology, i.e. all the linguistic fields of both languages. This implied that the affinity between Hebrew and Finnish was demonstrated in an all-round shape which followed the current traditions and principles of the scholarly research of his time.

On the basis of this argumentation it is logical to conclude that the

[21] See Heikel 1894: 56-58, and Salminen 1978: 240-241, 245-248.

[22] On the grammatical theories of that period, see G.A. Padley, *Grammatical Theory in Western Europe 1500-1700*. Trends in vernacular grammar, I-II (Cambridge 1985, 1988); Esa Itkonen, *Universal History of Linguistics : India, China, Arabia, Europe* (Amsterdam studies in the theory and history of linguistic scince. Series 3, Studies in the history of the language sciences, Vol. 65, Amsterdam 1991).

The article "Suomen kielen kuvaus 1600-luvun kieliopeissa" by Sakari Vihonen *(Collegium scientiae*. Suomen oppihistorian kehityslinjoja keskiajalta Turun akatemian alkuaikoihin. Editor: Jussi Nuorteva. Suomen Kirkkohistoriallisen Seuran toimituksia 125, Helsinki-Saarijärvi 1983: 121-155) includes a fine presentation of the philological literature known by the scholars at Academia Aboensis in the seventeenth-eighteenth centuries.

For the dissertations dealing with Oriental studies defended at the Academia Aboensis, see the catalogue in the article "Lähteitä orientalistiikan ja Vanhan testamentin eksegetiikan historiaan 1640-1828" published by Klaus Karttunen, in: Ilkka Antola & Harry Halén (toim.), *Suomalaisen eksegetiikan ja orientalistiikan juuria* (Suomen Kirkkohistoriallisen Seuran toimituksia 161. Helsinki 1993: 163-202): 163-179.

search for the roots and relatives of the Finnish language, which took place in the seventeenth century in academic circles, constituted a part of serious and consequent philological or linguistic research; it was not merely a capricious peculiarity intended to invent a glorious past for one's ethnic group. Even in those days prophecy was a rare phenomenon among scholars, and thus our predecessors could not predict the achievements of comparative linguistics which from the second half of the eighteenth century on was directed along completely new lines. Before that the biblical story of the confusion of languages at the tower of Babel constituted an axiomatic explanation of the variety of languages of the world. In this sense it was not illogical to search for vestiges of the pre-confusional language (as a rule considered to be Hebrew)[23] retained in various languages. A high number of such features could be interpreted as testifying in favour of a special relation with the Holy Tongue, and even a kind of competition can be seen to have taken place in this field. In another article I have referred to a number of parallel word lists which were collected by Sebastian Münster (1489-1552), Sveno Jonæ (died 1642), Olav Rudbeck *junior* (1660-1740), and Eberhard Gutsleff *junior* (1732) with regard to the similarities between Hebrew and German, Swedish, Lappish (Sami), and Estonian, resp.[24] Pierfranceso Giambullari (1495-1555) represents an additional parallel case in his book *Il Gello* (Firenze 1546) in which he refers to Hebrew in order to explain the origins of the Tuscan-Italian language of Florence. I am convinced that the number of these languages supposed to be related to Hebrew could easily be increased by numerous others through a review of the philological literature of the sixteenth-eighteenth centuries.

In Finland this type of research was continued during all of the eighteenth century. Daniel Juslenius (1676-1752), Professor of the (Holy) Languages (1712-1713, 1722-1727) and Theology (1727-1734) in Turku, Bishop of Porvoo / Borgå in Finland (1734-1742), Bishop of Skara in Sweden (1744-1752), a scholar of Finnish history and language, and the most well-known Fennophile of his time, dealt with the relation of Finnish to Hebrew in several publications (his dissertation *Aboa vetus et nova,* 1700; *Vindiciæ fennorum,* 1703; the inauguration speech *De convenientia lingvæ Fennicæ cum Hebræa et Græca,* 1712/1728; the introduction to his Finnish-Latin-Swedish dictionary *Suomalaisen sana-lugun coetus,* 1745).

[23] In contrast to the view of a number of "progressive" scholars, this was the conviction of Svenonius (see Salminen 1978:245-248, 256-257), and it was repeated by his followers, e.g. Daniel Lund (1692: 3).

[24] Tapani Harviainen, 'Ragaz ja rakas. Kai on suomikin heprean sukua?' *Kirjoja ja muita ystäviä.* Onnittelukirja Kaari Utriolle ja Kai Linnilälle (Toimittanut Marjut Paulaharju Karisto Oy, Hämeenlinna 2002): 69-74.

Juslenius was an energetic proponent of the honourable status of the Finnish language who concluded that Finnish was one of the independent cardinal, i.e. basic, languages which, in turn, had given rise to Lappish, Estonian, and Bjarmian, perhaps also to the Slavonic language. The origin of Finnish was to be derived from the Babylonian confusion of languages, and thus "no other language can boast of having given birth to Finnish"; the vestiges of Greek and Hebrew constitute only a part of the Finnish language.[25]

However, in his professoral inauguration speech *Oratio de convenientia linguæ fennicæ cum hebræa et græca* at the Academia Aboensis in 1712 Juslenius stressed the affinities of Finnish with Hebrew (and Greek) as a proof of the importance of the Finnish language.[26] The lexical contacts were described by him in the form of a score of striking equivalents (four of them occurred in earlier lists), though, according to him, there occur six hundred similar ones and, in addition, countless others which by form or reference are more remote but surely related, however. In the future Juslenius wished to return to these counterparts.[27] The words selected by Juslenius for his speech can be seen below (the transcriptions occur only in his manuscript):

Exclamatory אבוי awoi, avoi ('alas!') = woi.
Exclamatory אהה ahah, ahah = Finnish ahah.
זה (ze, 'it, this'; the transcription is lacking in Juslenius) = se.
נערה naara, na'ara; puella ('a girl') = naara.
אח ach, ach; focus ('a fireplace') = ahjo.
איש isch, ish; vir ('a man') = isæ ('a father').
אם em, em; mater ('a mother') = emæ, æmmæ vetula ('a mother', 'an old woman').
תאלה & אלה alah, ala; taalah, ta'ala; juramentum ('an oath') = wala.
חזה chasa, chaza ('he had a look') vel infinitivum חזה kheso, chezo; videre

[25] *Aboa vetus et nova,* Diss., Academia Aboense, Moderatore Joh. Berhn. Munster, [Aboa 1700]: II:2, III:33.

[26] In 1728 the speech appeared in an abbreviated version (called *Dibre chanukka* in Hebrew) in *Schwedische Bibliothec,* I (published by Chr. Nettelbladt, Stockholm 1728: 157-168); however, a complete manuscript of the speech is kept in the Helsinki University Library, call number A III 80. For Juslenius' opinions, see also Aarne J. Pietilä's doctoral dissertation *Daniel Juslenius - hänen elämänsä ja vaikutuksensa* (Tampere 1907): 146-154.

[27] "... ad oculum oriri patet; & quæ quærenti sexcenta occurrunt; præter quæ sono vel significatione aliquantum sunt remotiora, certæ tamen affinitatis innumera, sed jam consulto omissa, aliiqve occasioni, si pacem & vitam concesserit *hòs hypértata ãõmata naíei,* reservanda. Nunc vero plura eadem brevitate attingemus" (Juslenius' manuscript: 2-3; Juslenius 1728: 160).

('to see') = katzo. Chaldæorum inde ortum חזוא chaso, chezwa; aspectus ('appearance, apparition') transit in nostrum kaswo facies ('a face').

טבח tabach, tabach; occidere ('to kill') = tappa.

ילך jalach, yalach; ivit ('he walked') = jalca pes ('a foot, leg').

יעה jaah, ya'a = ajaa ejicere ('to drive').

כלא chylla, kulla;[28] omne, totum ('wholly', 'totally') = kyllæ (= kyllä/in), satis ('sufficiently').

מדה middah, midda; mensura ('a measure') = mitta.

מדד maddad, madad = mitata ('to measure').

קריה *(sic pro* קריאה) kirjah, kirya; lectio ('reading') = kirja liber ('a book').

רבץ rawaz, ravats; accubuit ('it lay down', sc. to eat) = ravitze saturavit ('he fed').

רגז, רגזת ragas ragsath (?), ragaz ragzat (?); commoveri affectu ('to be moved by affection') = racas, racasta dilectus, diligere ('beloved', 'to love').

Although we know that the comparative word lists consist of casual similarities, we may pay attention to the remarkable difference between Svenonius' list and those of his followers inclusive of the one collected by Collin, to be presented below: very few of the equations proposed by Svenonius were repeated by later scholars; instead they were able to find a rather large number of other pairs of words which indeed looked very convincing from their viewpoint. In my opinion, this indicates that, while the basic idea of Svenonius was considered to be correct for a long time after his death, his comparative material was estimated to be defective, irreliable, and perhaps even ridiculous in the view of other scholars who themselves were native speakers of Finnish. In this sense the development of the comparative lists also reflects a constant attempt to amend the quality of the argumentation in favour of the affinity between the two languages.

After the comparison of vocabulary Juslenius returned in his speech to the same morphological, syntactical, poetical, and orthographical categories which were earlier presented by Eric Cajanus (see above). In comparison with Cajanus' achievements, Juslenius was also able to pay attention to several new similarities in the field of morphology: the pronominal suffixes of the first person singular are *-i* and *-ni* in Hebrew and -ni in Finnish; in both languages the difference between singular and plural nouns with a pronominal suffix consists of a change in the vowel between the noun and the suffix (however, in Finnish only in the "accusative"), e.g. *debar**i*** vs. *debar**ay*** = San**ani** vs. San**ojani** ('my word' vs. 'my words'); similarly (a

[28] Ezek. 36,5.

preposition and) a pronominal suffix can be added to (infinitives of) verbs, e.g. *be-bhorch**o*** in fugere eum / cum fugeret = paëtesa**nsa** ('when he fled') and *be-qor'* ***i*** = rucoillesa**ni** ('when I prayed'); a "particle" (preposition) can be added with personal suffixes, e.g. *neged* coram ('in front of'): *negdi - negdecha - negdo* coram me/te/eo = edesæni, edesæs, edesæns ('in front of me/you/him') etc.; also the fourth "conjugation", i.e. the reciprocal *Hithpaël* (stem) of Hebrew verbs has a counterpart in Finnish, e.g. *hitgallel* / *hitgalgel* = kierin ('he / I rolled him/myself'). This demonstration of the affinity between Hebrew and Finnish is followed by a description of the parallels which in Juslenius' opinion connect Finnish with Greek.

Juslenius became a central figure in the cultural life of Sweden and Finland in the first half of the eighteenth century. Thus his special role in the history of supposed Hebrew-Finnish connections was to plant this conception in the minds of a rather extensive readership who at that time were increasingly interested in the glorious past of the Finnish people. As a consequence, Daniel Juslenius is the person who as a rule is later referred to when this Hebrew "track of errors" is mentioned.

A century after Svenonius' studies, on November 26, 1766, Fridericus (Fredrik) Collin (1743-1816), later (1784-1816) vicar of the parish of Helsinki, published the second part of his *pro gradu* (magister) thesis *Dissertatio historica de origine Fennorum* (p. 27-46) at the Academia Aboensis in Turku;[29] the *præses* of the disputation was Johannes Bilmark, the Professor of History and Practical Philosophy. Collin was born in Ruovesi, in the Finnish-speaking province of Häme. He completed his theological and humanistic studies at the Academia Aboensis and was rector of the Grammar School in Hämeenlinna / Tavastehus from 1775 on till his appointment in Helsinki in 1784.[30]

As a methodology to demonstrate his thesis of the Hebrew- Finnish affinity (convenientia), Collin first refers (p. 33) to the material features,[31] i.e. to numerous similar words with similar "root characters" (i.e. consonants) in both languages. However, only the similar references of these similar words can serve as evidence in favour of the relation; the root characters

[29] The first part was presented at Academia Aboensis on June 2, 1764 (4+26 pp.;Vallinkoski, Nos. 270-271). Collin considered that a number of Jews deported from Israel and Judah to Assyria and Babylonia moved together with Scythians to the North, where they became ancestors of the Finns; similar habits and customs in addition to the linguistic similarities served as proofs of this hypothesis.

[30] For Collin, see Herman Hultin, *Helsinge församlings historia* (Helsingfors 1930): 48-49, and Eeva Ojanen, *Helsingin pitäjän seurakunnan historia* (Helsinki 1972): 117-118, etc.

[31] As for the terms "material" and "formal" (see below), Collin refers to Guiljelmus Vottonus and his Dissert. Philolog. ad Chamberlaine.

can vary according to certain rules, however. Second, it needs to be demonstrated that as root characters the consonants are more essential in both Hebrew and Finnish; in contrast, the vowels can vary and transform the reference of words in innumerable ways.[32]

As a demonstration Collin presents (p. 30-33) a list consisting of 77 Hebrew words with their Finnish counterparts which fulfil his aforementioned prerequisites. Collin admits (p. 33) that he himself did not find all of these parallels; a number of them were presented by his predecessors (Daniel) Juslenius, Eric Cajanus, and Olaus Rudbeck; their literary notes were supplemented by oral information provided by (Anders) Lizelius, Dean of Mynämäki parish.[33]

In the period of Collin Hebrew was still included in general education, and thus every learned man was supposed to know the Holy Tongue fairly well. As a consequence, Collin could present the Hebrew words without vowels (which, completely correctly, were maintained by him to be of a minor significance). In favour of the readers of today – as was the case with Svenonius' Hebrew above – I have added to his list below a transcription after every Hebrew entry as well as English translations of the explanations given by Collin in Latin. In this list, too, underlining is added to point out the Finnish words.

This is the list of 77 words provided by Collin:

איש ish; *Hebr*. Vir ('man'), Isä *Fenn*. Pater ('father').
אב ab / av; Pater ('father'), in constr. אבי, abi; Appi Socer ('father-in-law').
אהה ahah; Aha ah! vox exclam.
אנה ana; Anoa & Anon obsecro ('I beseech, beg')
ערק arak; fugit ('he escaped'), Arca pavidus ('timid').
עשה asa; fecit ('he made'), Ase instrumentum ('instrument').
אור or; lux, Sol ('light', 'sun'), Auringo ('sun').
אי e, ey; Ei ('no', 'not').
אם em; Emä & Ämmä mater & vetula ('mother' & 'old woman').
אין en; non, En non ego ('I not', 'not me').
אות ot; Aawet signum, portentum ('sign', 'portent, prodigy').
אבוי avoy; Woi væ ('alas!').

[32] In fact, the latter thesis concerns a typical phenomenon in the Semitic languages which has been known from the very beginning of Semitic studies.

[33] Anders / Antti Lizelius (1708-1795) was a well-known publisher and journalist in Finnish and a *primus motor* of the new Finnish translation of the Bible published in 1758 and in revised form in 1776.

אח ach; Ahio focus fabrilis (‘a smith’s fireplace’).
תאלה & אלה ala & ta’ala; exsecratio (‘imprecation, curse’), Wala jusjurandum (‘oath’).
חלק chalak; partitus est (‘was divided’), Halki fissum (‘cloven’).
חמס chamas; vim intulit (‘he treated violently’), Hammas dens (‘tooth’), Hammastan mordicus impeto (‘I assail with the teeth, by biting’).
האביד he’evid; *Hiph*. punire fecit (‘he put/made to punish’),[34] Häwittä perdere (‘to destroy’).
הלל holel / holal; insanivit (‘he went mad’), Hullu insanus (‘folly, infatuated’).
הללות holelut; Hulluus, stultitia (‘stupidity, folly’). הוללה holela, idem.
המון hamon; strepitus (‘noise’), Humina Sonus venti (‘the sound of wind’).
ילך yalach; ivit (‘he walked’), Jalca (‘foot’, ‘leg’).
כלי keli; utensile (‘utensil’), Calu res, supellex (‘thing’, ‘set of articles, outfit’).
כלות kelot / kallot; terere (‘to use up, wear out’), Calutan rodor (‘it is gnawed, nibbled’).
קמט kamat, corrugare (‘to crumple up, shrivel’), Cammotta caveri (‘to beware of’).
קפא kapha; condensare (‘to make hard/firm, condense’), Capia arctus (‘firm, narrow’).
חרש charash; fabricatus est (‘it is forged’), Caraisen induro ferrum (‘I steel iron’).
חזה chaza; vidit (‘he saw’), Catzon video (‘I see’).
חצי chatsi; dimidium (‘half’), Caxi duo (‘two’).
גלל galal; volvit (‘he rolled himself’), Kelaan conglomero funem (‘I wind a rope’).
קרא kara; clamavit, oravit (‘he shouted’, ‘prayed’), Kerjätä mendicare (‘to go begging’).
כלא *in Piel* kille; prohibuit (‘he forbade’), Kieldää negare (‘to deny’).
קהל *in Piel* kihel; convenit (‘he convened’), Kihlata despondere (‘to betroth’).
חיל chayil; strenuitas (‘activity’), Kiltti egregius (‘excellent’).
חרב cherev; gladius & quodvis instrumentum consumtionis (‘sword & any instrument of consuming’), Kirwes securis (‘axe’).
חכות chakkot; expectare (‘to wait’), Cocotan expecto (‘I wait’).
חרב charav / charev; arescere (‘to become dry’, ‘to dry up’), Corwetan ustulor (‘I scorch’).
כלה *in Pyal* kulla; teri (‘to wear away’), Culun atteror (‘I wear away’).
כן ken; sic (‘so’), Cuin sicut (‘just as’).

[34] Perhaps the Latin counterpart is an error; the common Hebrew verb means ‘to destroy’.

כמר kamar; contraxit ('it became tighter, contracted'), Cumarran flecto me ('I bow down').
קבה kebha; cavum ventris ('the stomach cavity'), Cupu ingluvies avium ('the crop of a bird').
חמם chamam; calidus fuit ('was hot'), Cuuma fervidus ('hot').
כלה kullo; totum ejus ('its totality', 'all of it'), Kyllä satis ('sufficiently').
אדמה adama; Maa terra ('land, soil, earth').
מדה midda; Mitta mensura ('a measure').
מעט me'et, *in Piel* exiguus est ('is minor, scanty'), Mieto tenuis ('mild, light, weak').
מזה miz-ze; Mistä a quo sc. loco ('whence', i.e. 'from which place').
נערה na'ara; puella ('a girl'), Naara puella prostratæ pudicitiæ ('girl of prostrated chastity').
נכה nakha; percussit ('he stroke'), Nacka abjicere ('to throw away').
נעם na'am; amœnus fuit ('was charming'), Namu cupidiæ (= cuppediæ) ('dainty dishes, tidbits').
נוח nuach; quiescere ('to rest'), Nuckua dormire ('to sleep').
פלה pala; separavit ('he separated'), Pala frustum ('a piece').
פקוד pakod; mandare ('to order, command'), Pacottaa cogere ('to compel').
בן ben; filius ('son'), Penicka catulus ('whelp, puppy').
פתה pata / pote; improvidus ('improvident, apt to be deceived'), Petettää seduci, falli ('to be seduced, misled, deceived').
פלאות pela'ot; occultare ('to hide'),[35] Pilata (= pilailla) illudere ('to jest').
בנה binna; *In Piel* exstruere ('to pile up, construct'), Pinota struem conficere ('to prepare a pile').
בוח (an error *pro* פוח) puach; locutus est ('was spoken'), Puhua loqui ('to speak').
רבץ ravats; accubuit ('it lay down', sc. to eat), Rawitsen saturo ('I feed').
רגזה rogza; commotio ('a motion'), רגז ragaz; commotus fuit ('was moved'), Rakas / Rakasta dilectus, diligere ('beloved', 'to love').
רכב rekhev; currus ('chariot'), Reki traha ('sledge').
ריק rek; inane ('empty, void'), Ricka minimum quid ('a minimum quantity of something').
רנה rinna; cantus ('song'), Runo carmen ('song').
זה ze; Se ille ('this, it').

[35] Obviously an error, since *pela'ot* is a plural noun referring to 'miracles, miraculous events'; the dictionary by Ganander (see below) does not offer a Hebrew counterpart of this verb pilaan - 'I jest' (Ganander 1997: 704), either.

צמאה & צמא tsama & tsim'a; sitis ('thirst'), Siemen (= sieme, siemaus)[36] haustus, potus ('draught', 'drink').
דגן dagan; frumentum ('corn, grain'), Taikina massa ('dough').
דעת da'at; scire ('to know'), Taito / Tietä scire ('skill' / 'to know').
דכא dakha; contundere ('to beat, crush, squeeze together'), Takoa tundere ('to beat, hammer, strike').
תלל talal, accumulavit ('he heaped up, accumulated'), Tallillan (= tallataan / tallaillaan) concutior ('it is pressed together').[37]
תהו tohu; inane ('empty, void'), Tyhjä inanis ('empty, void').
ארח orach; via tecta ('a paved/treated way'), Ura via nive tecta ('a way paved/treated in the snow').
בעל ba'al; dominatus est ('he is ruler'), Walda / Wallita potentia ('might, power' / 'to rule').
טמא tame; inquinatus fuit ('he was polluted'), Tahmia lentore inquinare ('to pollute with a sticky substance').
יעה ya'a; ejecit ('he drove'), Ajan urgeo, pello ('I urge', 'I drive').
קריאה keri'a; lectio ('reading'), Kirja liber ('a book').
הביש hevish; pudefacere ('to make ashamed'), Häwäisen pudore suffundo ('I pour shame upon').
שדד shadad; bellum gerere ('to carry on a war'), Sodin bellum gero ('I carry on a war').
שלל shalal; spolium ('booty, spoil'), Saalis præda ('prey, booty').

A comparison with Svenonius' list clearly indicates that Collin paid strict attention to his propositions, which demanded similarity of both the consonantal structure and the reference of the words. In this sense he did demonstrate the correctness of his hypothesis. In addition to this "material similarity", he repeated once again the aforementioned morphological, prosodic, and syntactical, i.e. "formal features"[38] which also according to Daniel Lund, Cajanus, and Juslenius connected Finnish with Hebrew (p. 33-35). Parallels in material culture, manners and customs,[39] were added to

[36] Cf. Ganander 1997: 867 where the same Hebrew counterpart is mentioned in connection with the word sieme 'a draught of potion'.

[37] Cf. tallaan and tallailen in Ganander's Dictionary (1997: 948) which are connected with the Hebrew verb *talal;* in fact these Finnish verbs mean 'to tread, stamp (underfoot)'.

[38] See above, note 31.

[39] Though I have expressed above my dislike for the attempts to describe the achievements of our predecessors in a ridiculous light, a connection proposed by Collin between the unleavened Passover bread of Jews and the Finnish Easter pudding *mämmi*, the name of

the chain of evidence. In theory, the close connection between Hebrew and Finnish was now demonstrated as multi-laterally and convincingly as the paradigms of the current philology could ever demand.[40]

The word list of Collin was to receive a permanent position in the study of Finnish, when it was included by Christfrid Ganander, pastor of the parish of Rantsila (1741-1790), in his extensive dictionary of the Finnish language; in contrast to the earlier lists Collin's achievement was obviously considered to be most reliable. *Nytt Finskt Lexicon* was completed by Ganander in manuscript form in 1786-87; in it the author offers Hebrew etymologies and/or counterparts for almost one hundred Finnish words without mentioning the source of these notes. Nevertheless, the identical spelling mistakes, printer's errors etc. in Ganander's Hebrew indicate that he in fact copied the whole list of word comparisons collected by Collin in his dissertation; a few other words were added from other sources. Unfortunately, however, the dictionary by Ganander did not appear in print earlier than

which clearly originated from Hebrew, probably transformed into Finnish from the biblical celestial bread *manna* (p. 43-44), is too amusing to be passed by without a note. In part, Collin has taken the reference to *mämmi* from Daniel Juslenius, who in the *Aboa vetus et nova* (1700, III: 28) and in his Dictionary (1745) wrote that *mämmi* is eaten in Turku at Easter in memory of the unleavened bread.

[40] Still later, these "formal" arguments were repeated by Carolus Gustavus Weman (1740-1803) and his *respondent* Benedictus Jac. Ignatius in the *De convenientia linguarum hebrææ et fennicæ,* a dissertation defended at the Academia Aboensia in 1767 (Vallinkoski, No. 4276), although according to Weman (p. 16), Collin had demonstrated the affinity both "in materialem & formalem" in his dissertation. As for the vocabulary, however, Weman was satisfied with a quotation of Henricus Ganander (p. 13), who in his grammar of Lappish published in 1743 (in fact, it is an open question whether this grammar was ever published, cf. Nuutinen, in: Christfrid Ganander 1997: xi) had offered the following six comparisons which in his opinion are shared by Hebrew, Lappish, and Finnish:

צנף tsanaf, circumligavit	Zianam ligo	Sidon
צינק tsinok, Nervus	Suodnac funis ex nervis	Suonicko
צלם tselem, Imago	Zialbme oculus	Silmä
ילך yalakh, ambulavit	Juolka pes	Jalka
יד yad, manus	Kiedta manus	Käsi
ידה yada, manavit, civit	Jodam profiscor	Judan

A footnote by Weman shows that he did not understand Judan, a north-Finnish loan from Lappish, because he wishes to correct it to *Joudun, pervenio, pergo celeriter. Forte etiam nomen Juhta jumentum huc referri potest*. Manavit and civit mean 'to flow, spread, move, stir', and both the Lappish *Jodam* and Finnish Judan 'I journey, travel'.

On p. 14-15 Weman refers to (Olav) Rudbeck *filius* according to whom the Finns originated from the ten lost tribes of Israel.

1997.[41]

However, only a few decades after Collin and Ganander the study of comparative linguistics was to acquire a totally different direction under the leadership of Wilhelm von Humboldt, Franz Bopp, Rasmus Rask and Jacob Grimm. Our predecessors could not predict future developments. As part of the European community of scholars they followed the scholarly paradigm of their own period.

[41] A scholarly edition: Christfrid Ganander, *Nytt Finskt Lexicon*, ed. by Liisa Nuutinen, I-II, Helsinki-Hämeenlinna 1997.