ON THE USE OF NATURAL LANGUAGE PROCESSING FOR AUTOMATED CONCEPTUAL DATA MODELING

by

Siqing Du

Master of Science, Tianjin University, 2003

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2008

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Siqing Du

It was defended on

June 25th 2008

and approved by

Dr. Douglas Metzler, Dept. of Information Science & Telecommunications

Dr. Rebecca Hwa, Dept. of Computer Science

Dr. James Joshi, Dept. of Information Science & Telecommunications

Dr. Hassan Karimi, Dept. of Information Science & Telecommunications

Dr. Vladimir Zadorozhny, Dept. of Information Science & Telecommunications

Dissertation Director: Dr. Douglas Metzler, Dept. of Information Science & Telecommunications

ON THE USE OF NATURAL LANGUAGE PROCESSING FOR AUTOMATED CONCEPTUAL DATA MODELING

Siqing Du, PhD

University of Pittsburgh, 2008

This research involved the development of a natural language processing (NLP) architecture for the extraction of entity relation diagrams (ERDs) from natural language requirements specifications. Conceptual data modeling plays an important role in database and software design and many approaches to automating and developing software tools for this process have been attempted. NLP approaches to this problem appear to be plausible because compared to general free texts, natural language requirements documents are relatively formal and exhibit some special regularities which reduce the complexity of the problem. The approach taken here involves a loose integration of several linguistic components. Outputs from syntactic parsing are used by a set of hueristic rules developed for this particular domain to produce tuples representing the underlying meanings of the propositions in the documents and semantic resources are used to distinguish between correct and incorrect tuples. Finally the tuples are integrated into full ERD representations. The major challenge addressed in this research is how to bring the various resources to bear on the translation of the natural language documents into the formal language. This system is taken to be representative of a potential class of similar systems designed to translate documents in other restricted domains into corresponding formalisms. The system is incorporated into a tool that presents the final ERDs to users who can modify them in the attempt to produce an accurate ERD for the requirements document. An experiment demonstrated that users with limited experience in ERD specifications could produce better representations of requirements documents than they could without the system, and could do so in less time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

First I would like to thank my academic advisor Dr. Douglas Metzler for his tremendous support and insightful guidance over the past five years. I sincerely appreciate the help and encouragements he provided for me during my Ph.D study. He not only let me develop the research topic I am interested in, but advised me whenever I encountered difficulties during the whole process of this undertaking. Without his numerous proof reading of my rough drafts and invaluable suggestions for rewriting, this thesis work would be far worse. I owe limitless gratitude to him.

I would also like to thank the members of my thesis committee: Dr. Rebecca Hwa, Dr. James Joshi, Dr. Hassan Karimi and Dr. Vladimir Zadorozhny, for their extremely thoughtful inputs and examinations of this work on different levels from different angles applied with their expertise. I am immensely grateful and deeply indebted to them.

This project also involved many open source packages such as WordNet, Link Parser, Stanford NLP parser, Graphviz, etc. Thanks to those people who made these invaluable resources freely available. Developing an automated conceptual data modeling system would not be possible without the availability of these open source packages.

I would also like to acknowledge the generous financial support and assistance provided for me by the school of Information Sciences, University of Pittsburgh.

Finally, thanks to my family unconditional support, care and encouragements. They are indispensable for me to accomplish this work. I could never have enough words to thank them for what they did for me.

1.0 INTRODUCTION

1.1 MOTIVATIONS AND PROBLEM STATEMENTS

This research is motivated by both the importance of automated conceptual data modeling in the design of information systems and databases (ISs/DBs) and the emerging applications of natural language processing (NLP) in information retrieval and extraction.

Conceptual data modeling, which aims at exploring high-level data oriented structures, is one of the most important phases in the design of ISs/DBs. It not only provides a blueprint for the whole system, but determines most of the system functions, forms and structures. The importance of conceptual data modeling is also indicated by the fact that most of the database and software design textbooks have chapters on teaching people how to transfer requirements specifications to conceptual data models.

Automating the process of transforming requirements specifications to conceptual data models can provide significant benefits. Conceptual data modeling is a labor-intensive and time-consuming process. What makes the situation worse is that modifications of the user requirements are almost inevitable during the ISs/DBs development process [TF82], so conceptual data modeling has to be performed frequently, even during the development of a single application. Sometimes, due to the cost and time factors involved in re-construction of the conceptual data models, designers modify the conceptual models directly, instead of making modifications to the original requirements specifications. Hence, inconsistency can be introduced between the formal models and user requirements documents, which not only causes serious problems in the system maintenance phase, but also affects the reusability and prototyping of new systems with similar requirements [Dal92].

Currently, there are many data modeling tools available to aid or automate the process

from conceptual data models to logical designs and to physical implementations, such as ERWin from Computer Associates, Visio from Microsoft, Oracle Designer from Oracle, and Dia from Gnome Office. Most of them offer not only forward engineering from conceptual data models to logical models, and to physical models, but also reverse engineering from physical implementations to logical models, and to conceptual data models. However, there is still no general tool available for transforming natural language requirements specifications to conceptual data models.

Automated conceptual data modeling is essentially a translation process – from natural language representations to some kind of formal and structured representations. Natural language is the most often used language to represent the initial requirements. It was found that between 93% and 95% of all the user requirements in industrial practice were written in natural language [NL03, MFNI04]. On the formal and structured representation side, there are many different target formalisms for conceptual data modeling in different domains. For instance, Entity Relationship Diagram (ERD) is often used for database design while Class Diagram is frequently used for object oriented software design.

Many challenging problems are involved for automated conceptual data modeling. How to recognize and extract the important concepts in a requirements document is one of them. Conceptual data modeling is traditionally performed by information analysts and many implicit decisions are made during the modeling process. One of the important tasks of conceptual data modeling is to extract the important concepts from the requirements documents; for instance, "employee" and "project" in "a number of employees work on each project". The recognition of important concepts is an implicit process and may be based on the background and world knowledge of the information analyst. It is a notoriously difficult problem to formalize the general background and world knowledge of human beings. Heuristic rules based on syntactic, semantic or pragmatic information have been proposed in some of the previous research. For instance, Chen [Che83] proposed "A common noun in English corresponds to an entity type in an ER diagram". These heuristic rules are very useful and provide general guidelines to how to extract the important concepts. However, they are very general and incomplete, much more fine-granulated heuristic rules are needed for a relatively accurate and practically usable conceptual data modeling system. The past

two decades have seen some initiatives [Che83, TB93, BCD+95, OHM04] in this direction. However, most of the previous heuristic rules for transforming natural language requirements to formal representations were based on word classes and other low level information. Even the most recent one [OHM04] utilized only shallow parsing and chunking. Increasing usage of dependency parsing for various NLP tasks, from machine translation to question answering [dMMM06] were reported recently because of the high level grammatical relations offered between individual words. It seems that heuristic rules based on high level information such as grammatical relations from full level dependency parsing are very promising for formalism extraction in automated conceptual data modeling. So one of the specific goals of this research is to investigate and propose a relatively complete set of heuristic rules based on high level grammatical relations.

Handling the various ambiguities in automated conceptual data modeling is another challenging problem. Ambiguities can happen at morphological, syntactic, semantic and pragmatic levels. Although a large body of techniques, methodologies and tools are made available with the research and development in computational linguistics, the intended domains of these resources may be quite different from requirements documents. It is of great interest to study the applicabilities of the relevant technologies, tools, and resources to automated conceptual data modeling. Various domain and application specific problems need to be investigated, for instance, entity attribute disambiguation, entity relation disambiguation, attribute name extraction, specific database requirements language structures disambiguation, etc. The increasing availability of large-scale general knowledge resources and improvements in knowledge-based semantic analysis, which have not been used in the previous research, are promising resources that can be explored to improve accuracy in automated conceptual data modeling.

Automated conceptual data modeling also involves a knowledge representation problem. The natural language input and the desired formalism are two forms of surface representation of one common deep semantic representation – the conceptual data model of the requirements. Proper data structures need to be designed to hold the information and knowledge for the transformation between different representations.

Although it is still quite difficult to transform general free texts to formal representations

such as first order logic, the solvability of this automated conceptual data modeling problem is due to the regularities and sublanguage characteristics of the requirements documents. Furthermore, many of the NLP hard problems such as speech acts, agent belief and intention, are not involved in automated conceptual data modeling while the complexities of some other NLP hard problems may also be reduced. Moreover, there have been some successful systems in sublanguages similar to requirements documents.

In general, complete solutions to transforming natural language requirements documents to conceptual data models calls for knowledge and advances in several fields: Natural Language Processing (NLP), Knowledge Representation (KR), Machine Translation (MT), Information Extraction (IE), and Common Sense Reasoning (CSR).

1.2 BACKGROUND AND RELATED RESEARCH FIELDS

The following sections provide background knowledge on conceptual data modeling and a brief review of related research fields.

1.2.1 Conceptual data modeling (CDM)

The general steps to follow in the design of IS/DB systems are user requirements acquisition, conceptual data modeling, logical data modeling and physical data modeling. Conceptual data modeling is situated right before logical data modeling and after the requirements acquisition phase. The central problem of conceptual data modeling is to extract a set of entities and their relationships that represent or model a particular information aggregation specified in the requirements descriptions.

A four-phase requirements process model is illustrated in Figure 1, adopted from [KM95]. Users perceive the enterprise reality and provide requirements to information analysts. Information analysts interact with users to discover their explicit and implicit requirements and document the requirement specifications (usually in natural language). Then, information analysts (maybe different people) build formal conceptual data models

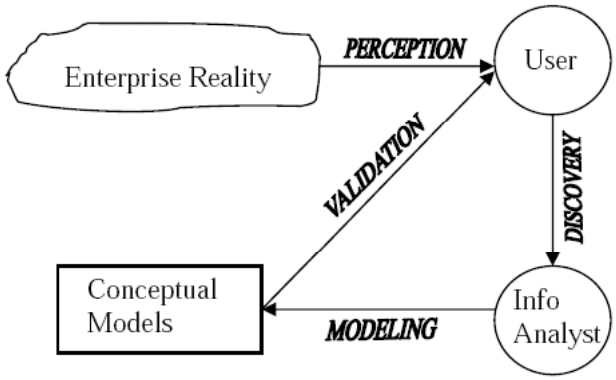


Figure 1: A process model of information requirements [KM95]

based on the information identified in the discovery process. These conceptual data models are usually the first formal representations that bridge together the end-users and the system designers. The loop of *discovery-modeling-validation* can iterate several times in order to get accurate conceptual data models.

In the past several decades, researchers have made great efforts on the formalization of requirement analysis. Various conceptual data modeling formalisms have been proposed such as Entity-Relationship (ER) modeling, Natural/Nijssen Language Information Analysis Method (NIAM), Object Role Modeling (ORM), Logical Data Structure (LDS), Unified Modeling Language (UML). A comparison of some of these data modeling formalisms in terms of quality of data model, quality of design time preference, time task performance, etc can be found in [KM95, NL03].

Among the different modeling methods, ER modeling and Object-Oriented modeling are extensively used in practice. In a market research study [MFNI04], 63% of the respondents who adopt methodologies were reported using ER data modeling and 68% reported using Object-Oriented methods. In the rest of this section, only these two categories of modeling formalisms are reviewed.

The ER model, introduced by Chen [Che76], is based on a natural view of the real world that consists of entities and relations. The basic elements of the ER model are entities, relations and attributes. Entity relationship diagram (ERD), the diagrammatic

tool proposed to assist the modeling process, is a widely used conceptual data modeling formalism for IS/DB designing [MFNI04]. One reason for its popularity is that ERD is easy to understand by both system analysts and end users. Many variants of ERDs were proposed by different researchers and utilized by different organizations. The E^2R (EER) model [EL89] and REMORA [Rol88] include features like multiple complex entities and attributes, generalization and specialization of entities, role concepts for relationship types, etc. Chen [Che81] presented a detailed discussion of the different variants of ERDs. Figure 2 is an example of entity relationship diagram following Chen's style.

Figure 2: An example of entity relationship diagram [Che81]

Object-Oriented modeling involves another important family of conceptual data modeling approaches, especially in software engineering. Unified Modeling Language (UML), which is developed by unifying primarily Booch, OMT, and OOSE methods [BRJ96], is officially defined by the Object Management Group (OMG) and now is becoming the de facto standard language for formal description of software requirements [BMO01]. Unlike ER modeling, which includes only one kind of diagram (ERD), UML 2.0 includes 13 types of diagrams: Class Diagram, Activity Diagram, Communication Diagram, Component Diagram, Composite Diagram, Deployment Diagram, Interaction Overview Diagram, Object Diagram, Package Diagram, State Machine Diagram, Sequence Diagram, Timing Diagram and Use Case Diagram. It is difficult to formulate rules to translate NL requirements specifications to all these diagrams. In fact, only the correspondence between NL components and Class Diagram has been investigated in previous literature.

1.2.2 Controlled languages and sublanguages

Controlled languages place special restrictions on grammar, style and vocabulary to the language used in special domains. Typically, a controlled language is formally defined, and the conformity to the controlled language standard can be verified. Several controlled languages are widely used in commercial applications. For instance, the International Aerospace Maintenance Language, which is designed for maintenance staff who may not be native English speakers, has been adopted by Boeing since 1990 [OM96]. The Attempto Controlled English (ACE) is a rich subset of standard English designed to serve as a specification and knowledge representation language [FS96]. Although ACE must be learned to be used competently, once the requirements specifications are written in ACE formats, there are many tools, such as Attempto Parsing Engine APE, Attempto Reasoner RACE, Attempto Verbaliser DRACE etc. that can be used to formalize the requirements specifications. The Caterpillar Fundamental English (CFE) and the Caterpillar Technical English (CTE) are controlled languages developed by Caterpillar in conjunction with the machine translation center at Carnegie Mellon University to simplify technical document authoring and translation. CTE contains about 8,000 general terms and 70,000 technical terms [Fie98, KAMN98].

Sublanguages are subsets of natural language used by a particular community of speakers with a particular subject matter or those engaged in a specialized occupation [BSA72]. Examples of sublanguages include weather reports [Isa84], software comments [EDB00], requirements specifications [Cyr95] and medical reports [SFL87].

Since the 1950s controlled languages and sublanguages have been investigated to simplify and reduce the complexities of some natural language processing problems, such as ambiguity and unknown words, in some specific domains. Controlled language and sublanguage methods have achieved success in systems developed for particular domains. For example, machine translations systems TAUM-METEO [Isa84] and ATR [Mor93] performed well in their domains: weather forecast reports and conference registration documents. In the medical domain, Sager & Naomi [SFL87] utilized sublanguage grammar analysis (mainly word class co-occurrence), semantic constraints (mainly in the form of selectional restrictions), both

during and directly after parsing to eliminate those syntactic analyses that give meanings impossible in various medical reports. In scientific and technical domains, Harris [Har89] argued that the initial phase of sublanguage analysis is important in establishing a direct relationship between surface sentence forms and the semantic representation.

1.2.3 Related research fields

1.2.3.1 Machine translation Automated transformation of natural language requirements documents to conceptual data models is essentially a machine translation problem. Automated conceptual data modeling and machine translation both require morphological, syntactical and semantic knowledge of the natural language, knowledge of the target formalisms, knowledge of the various correspondences between natural language and target formalisms and knowledge of general context and common sense [ABM+94].

The general approaches for machine translation are dictionary-based machine translation, interlingual machine translation, example-based machine translation and statistical machine translation [ABM+94]. The first two approaches usually involve building a large number of translation rules based on morphological, syntactic, and semantic processing. Some previous research in NLP-based conceptual data modeling has adopted similar approaches as dictionary-based and interlingual machine translation [RP92, GSD99]. The latter two approaches are usually based on the statistical information collected from bilingual corpora. The methods in themselves are application neutral and could be applied in conceptual data modeling if large scale corpora relating natural language and target formalisms existed. However, a large corpus of natural language requirements documents and their interpretations is not available currently.

The difficulty of dealing with automated high-quality translation of arbitrary texts has long been recognized [Kit87]. However, machine translation has been quite successful in some sublanguages, such as TAUM-METEO [Isa84] on weather forecast reports. Exploring automated language processing techniques in restricted formats and domains has a higher probability of success in the near future.

1.2.3.2 Information extraction Information extraction aims at filtering information from large volumes of texts [Gri97]. The DARPA sponsored Message Understanding Conference (MUC) and competition have greatly promoted research on information extraction in the past decade. A typical example is terrorist events extraction, which involves processing a set of unclassified newswire articles and determining the type of attack (bombing, arson, etc), the date, location, perpetrator and targets. The set of filled slots represents an entity with its attributes, a relationship between two entities or an event with various entities playing certain roles [Gri01]. The overall information flow is similar to that of automated conceptual data modeling except for the final target formalisms. They both involve analyzing the source texts by applying natural language processing and producing structured information. From this point of view, information extraction related techniques can be applied to the first several phases of automated conceptual data modeling, especially the syntactic and semantic analysis. With modeling domain knowledge, the output of an information extraction system may be transformed to a modeling formalism.

Typical information extraction applications are train or airplane travel information retrieval, car navigation systems, information desks [Gri97], event and relation extraction [CM00, HBR03] and terminological extraction [RDH+02]. The various techniques used, such as pattern matching and structure building [GLS95], Finite State Automaton Text Understanding and template filling [HAB+97], lexical and syntactic analysis [HBR03], name recognition [BMSW97], scenario pattern matching, co-reference analysis, inference and event merging [Gri97], are applicable to automated conceptual data modeling research.

As in the case of machine translation, the statistical and corpus-based approaches in information extraction have not been applied to automated conceptual data modeling yet because of the lack of generally available corpora for such research.

1.2.3.3 Text and web mining Text mining and web mining, especially web content mining [Fur04], two subfields of data mining, also involve analyzing large volumes of unstructured or semi-structured data and extracting interesting information. The research in these fields draws on approaches in data mining, information retrieval, machine learning, statistics, and computational linguistics, so some of the techniques and methodologies have

possible application in automated conceptual data modeling, especially the efforts on entity extraction, relationship and event detection, object identification, ontology and synonym discovery and reference resolution [MMS+02].

1.2.4 NLP in other phases of requirements engineering

NLP techniques have been applied in other phases of requirements engineering that are related to conceptual data modeling, such as the requirements elicitation and acquisition phase which precedes conceptual data modeling, and the requirements validation phase which follows conceptual data modeling.

Several benefits of using NLP in requirements engineering have been suggested, such as facilitating the digitizing of requirements documents with speech recognition or interrogation interfaces, detecting and revealing ambiguities and contradictions in requirements documents [RW91, HJL96], improving requirements document writing, and helping design interview questionnaires by detecting potential ambiguities in the questions [WRH97].

Some of the previous research tried to provide a general requirements engineering environment covering several phases of the IS/DB design process. The RADD [BCD+95] is a complex database design tool including a dialog tool for database requirements elicitation, a module for conceptual data modeling, and a component for database logical design. CIRCLE [AG97, AG06] is another general requirements engineering tool that provides both conceptual data modeling and requirements validation functionalities in terms of syntax and of semantics. Dalianis [Dal92] suggested a tool that utilized paraphrase, discourse analysis and natural language generation techniques for requirements validation. Gervasi & Nuseibeh [GN02] applied light-weight NLP methods, mainly shallow parsing, to check properties of models. Goldin & Berry [GB97] utilized lexical analysis to identify significant domain terms.

1.2.5 Non-NLP-based automated CDM alternatives

As stated in section 1.1, natural language is the most frequently used language for requirements documents and NLP-based approaches are the main approaches to automated

conceptual data modeling. However, non-NLP-based alternatives also exist. Several early approaches [SWL83, BDDL84, CMNK88] analyzed a collection of forms, instead of natural language requirements documents, to build schema diagrams. Dubois & Petit [DP95], Edwin & Jaco [EJ00] and Tsai *et al.* [TJS91] argued for the usage of formal and structured languages for requirements specifications because the precision of formal language is the most desirable feature for requirements specifications. The transformation of natural language requirements to some kinds of formal representations, such as UML and ERD, is also for the purpose of formalization of the requirements. However, approaches that skip the natural language requirements phase by requiring information analysts to write the requirements directly into formal representations involve much more human effort in the process and communication difficulty with the end users.

2.0 NLP-BASED CONCEPTUAL DATA MODELING

The past two decades have seen many attempts to solve, from various angles, the problem of translating informal descriptions (either controlled or sub-language texts) to formal specifications. In this chapter, the related issues, methodologies and evaluations methods regarding NLP-based conceptual data modeling in previous research are reviewed. See Appendix A for a chronology of the previous systems.

2.1 ISSUES REGARDING NLP-BASED CDM

The issues are discussed serially from input treatments, syntactic parsing, heuristic rule extraction, semantic analysis and disambiguation, domain and background knowledge, to output formalisms.

2.1.1 Input treatments

2.1.1.1 Preprocessing Because of the open-ended characteristics of text input, preprocessing is a necessary and effective way to remove noisy data and achieve normalization for information retrieval and extraction, machine translation and text mining. And it has been utilized in conceptual data modeling as well. The kind of preprocessing utilized in a system depends on the later processes involved in the system as well as the application domain. Researchers need to consider the nature of the inputs carefully and choose the proper procedures in order to deliver the desired formats to the transformation system. Sometimes it may be superficial. For example, if a specific parser utilized in

an application has trouble handling parenthesized explanations embedded in the text, a simple preprocessing that removes the parenthesized explanation will significantly improve the performance of the parser. Other situations may require relatively deep processing, such as canonicalization. Lee & Bryant [LB02a] preprocessed the text requirements into XML format by inserting syntax tags such as paragraph and functional tags such as preconditions. If the requirements documents are not well-formed, preprocessing that involves correcting, selecting and normalizing to reduce redundancy and inconsistency is usually used [Mic96].

2.1.1.2 Controlled languages The difficulty of dealing with general free text and the success of machine translation in controlled languages motivated some researchers to put constraints on the input, either by restricting the vocabulary or the sentence structures [BGM85, TB93, BvdR96, OM96, AG06]. With these restrictions, simple linguistic processing such as tagging and chunking can achieve reasonably good results in conceptual model building even without sophisticated semantic processing. This also enhances the tractability of many difficult problems in natural language processing such as ambiguity and unknown words.

An example of controlled language input for database requirements specification is shown in Figure 3, adopted from [BGM85]. All the sentences in this application had to be in *subject-verb-complement* form. Other variants were not allowed. A strict translation from such a description to a declarative programming-like middle language representation such as the following was performed:

EMPLOYEE : PERSON,

STUDENT : PERSON,

STAFF : EMPLOYEE, $\cdots$

Then this middle language representation was transformed to database schema. The limitation is obvious: this controlled language is unnatural and awkward. However, if the requirements documents are in such format, it is less difficult to handle them.

A similar but less restricted text, a scenario-based description, is shown in Figure 4, adopted from [BvdR96]. In this scenario-based description, simple sentences without subordinate clauses were used in order to simplify the processing. Most of the sentences

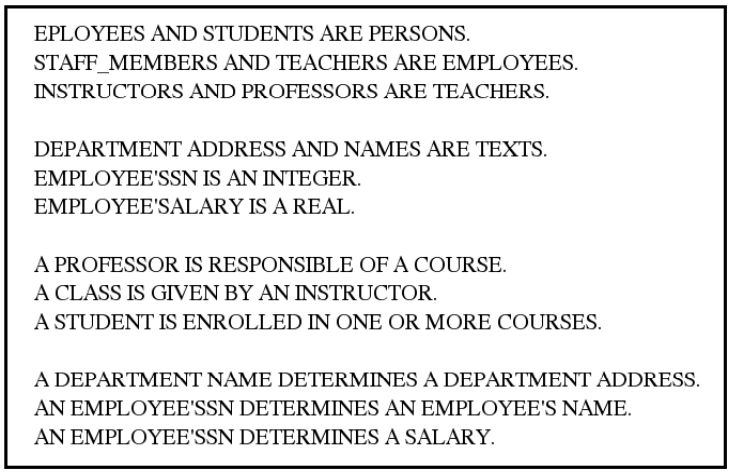


Figure 3: An example of controlled language input [BGM85]

were in the format:

$$< \text{subject} >< \text{verb} >< \text{direct object} > [< \text{indirect object} >]$$

The scenario descriptions were transformed into an event model in a semi-automated way which was supported by a lexicon. The transformation process involved defining the events individually aided by WordNet, and defining dependencies between events.

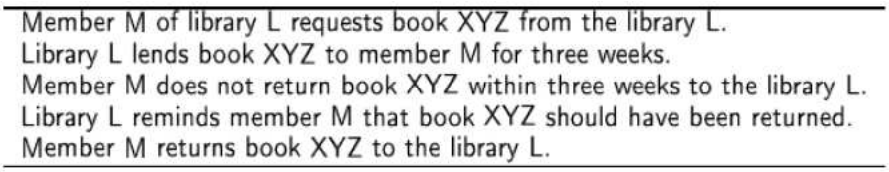


Figure 4: An example of scenario-based NL description input [BvdR96]

The use of controlled language methods is not without limitations. It does not apply to existing requirements documents. Conformance checking and transformation are needed. Furthermore, it is not natural and places burdens on the requirements documents writers.

2.1.1.3 Sublanguages General natural language requirements specifications can be treated as a sublanguage. For instance, the following database requirements statement which has been used as a classic example in several papers [BBLON77, TF80, Che83] [1] exhibits some sublanguage characteristics.

> The company has 50 plants located in 40 states and approximately 100,000 employees. Each plant is divided into departments and further subdivided into work stations. There are 100 departments and 500 work stations in the company. In each department there is an on-line time clock at which employees report their arrival and departure. A work task is associated with one of 20 different job types. Each of the job types can be performed at each of the plants. During a given day an employee may perform more than one work task, each associated with a different job type, and each can be performed at different work station. Each work station has an on-line data entry device at which an employee reports activity on a work task. There are five worker unions represented in the company, and every employee belongs to exactly one union. Although the size of the company remains stable, about 20 percent of the employees leave each year and are replaced by new personnel.

Lee & Bryant [LB02b] claim that requirements documents are easier to process than other types of textual documents in the sense that requirements documents usually have well defined structures with fewer ambiguities and infrequent use or narrow reference scope of pronouns. This kind of regularity can facilitate the syntactic processing of both general parsers [RP92, Fou99] and specifically-crafted parsers [GSD99] as discussed in section 2.1.2 on syntactic analysis.

2.1.1.4 Dialog Dialog-based tools are effective in requirements elicitation and acquisition. They are also used to guide the design process by enabling posting questions to designers [WR82]. Interactive user interfaces [BCD+95, HG03, OHM04] can bring users to bear in addressing the complexities of natural language processing such as disambiguation. Further processing can begin right after a sentence is input (either in controlled languages or sublanguage), instead of waiting for all of the text to be input. The system may post additional questions if it finds gaps in the input. For instance, the dialog tool used in [BCD+95] could post three types of questions: content questions (e.g. "Are there any more details about the application?"), linguistic clarification questions (e.g. "How is the act borrow done?") and pragmatic clarification questions (e.g. "How is book characterized?").

[1] This example text is used several times in this paper for various illustration purpose.

The major disadvantage of dialog-based systems is that more user interventions and attention are required. Hence it is hard to adopt dialog methods for large scale batch processing. However, a system can provide both dialog interface and batch processing as in the CM-Builder [HG03].

2.1.2 Syntactic parsing

Syntactic structure is an intrinsic part of natural language and the syntactic corresponding patterns between natural language and various target formalisms are important for automated conceptual data modeling. Chen [Che83] noticed the corresponding patterns between English sentence structures and ERD elements and proposed eleven high level heuristic rules to formulate these patterns (see section 2.1.3). Abbott [Abb83] and Booch [Boo86] discussed the corresponding patterns between programming language and natural language: data types vs. common nouns, variables vs. direct references and operators vs. verbs. Recently, Galatescu [Gal02] detailed the possible corresponding patterns among real world concepts, natural language elements, and UML as illustrated partially in Figure 5.

Research in NLP provides many syntactic analysis tools, including both special purpose and generalized Part-Of-Speech (POS) taggers and parsers.

The most often used parsers are constituent tree-based (phrase structures). Constituent trees represent the hierarchical nature of sentence constituents in a tree structure. For instance, if the input sentence is "the company has 50 plants located in 40 states", one of the parse trees is shown in Figure 6. Several levels of information can be derived from this constituent parse tree. The non-terminal directly above each word in the sentence is the POS for that word. Many of the context free grammar based parsers use the Penn Treebank POS tag set, which includes about 40 tags [Cha96]. There are also other tag sets such as, the Brown Corpus tag set which contains 87 simple tags, the Lancaster-Oslo/Bergen(LOB) Corpus tag set which contains about 135 tags, the Lancaster UCREL tag set which contains about 165 tags, and the London-Lund Spoken English Corpus tag set which contains 197 tags [MMBS93]. The sentence level information, such as sentence roles (also called grammar relations, such as subject, object) can be derived from the hierarchical relationship between

Real world	Natural Language	Unified Modeling Language
Complete assertion, attitude, question	Sentence	- Diagram/ Subdiagram
Complex idea, situation, application	Phrase/ Paragraph etc	- Complex diagram (e.g. Use Case or Package diagram)
Object, abstract notion, place, event, quality (simple/complex)	Noun (simple/ compound)	Object (flat/ composite)
Action, state, attitude, event (simple/ complex)	Verb (simple/ compound)	- Message (method/ event) - Activity/ Complex transition - Type of Association
Object feature, quality (simple/ complex)	Adjective (single/ multiple)	Object's attribute (value/ complex structure)
Action characteristic	Adverb	- Operation property - Attribute of the association
Anaphora (backward reference)	Pronoun / Anaphoric article	- Reflexive association - Object lifeline
Object – Action Relationship (Role of the Object in Action)	Preposition / Conjunction / Adverb (see Table 8)	- Role in association/ interaction - Relation between a message/ event and its parameters (objects)
Inter-Object Relationship	Conjunction (copulative/ disjunctive etc)	- Association/ Dependency - Aggregate composition - Message
Inter-Operation Relationship	Conjunction / Conjunctive adverbs	Simple/complex transition / Control icons (both in activity diagrams)
Object Plural	Cardinal of the individual noun/ Collective noun	- Object multiplicity in association - Multiobject
Object Quantification and Modality	Noun quantifier/ 'there is' construction	Object multiplicity in association
Action Modality	Modal verb	Decision symbols in activity diagram
Object Type–Instance dichotomy	Noun, Proper noun	object: Class
Specialized meanings of Objects/ Actions	Word Polysemy	Class specialization/ Interface
Encrypting the knowledge on Objects/ Actions	- Human Knowledge/ Metonymy/ Anaphora	Packing class definition /diagram
Disjunctive meanings/ possibilities of Objects/ Actions	Word Homonymy	Separate classes/ Decision among activities / OR split in transition/ Stereotypes
Question	Interrogative sentence	'Query' property of the operation
Embedded Objects/ Ideas / Actions	Embedded sentences	Composite objects/ Nested states

Figure 5: Corresponding patterns between NL and UML [Gal02]

the tags. For example, if a sentence is composed of NP and VP, the subject is usually the head noun of the NP.

The output structures of constituent tree based parsers were utilized in heuristic translation rules [TB93, BCD+95, OHM04]. For example, with the rule "Heuristic HE2: A common noun may indicate an entity type" [OHM04], "company", "plants" and "states" could be identified as three possible entity types from the parsing result generated from a Memory-Based Shallow Parser (MBSP) [DZBG96, ZD99] as in Figure 6. Lexical rules are not enough in this case, because both "plants" and "states" can be verbs.

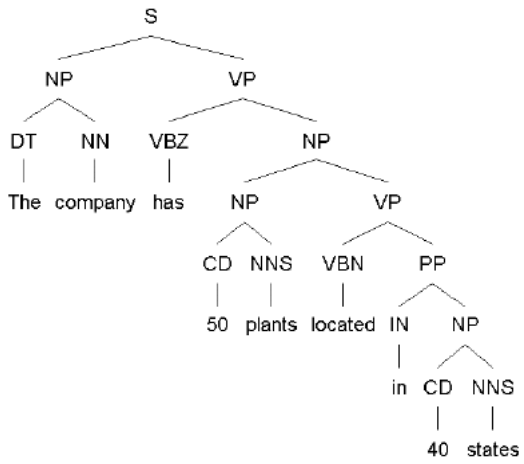


Figure 6: An example of constituent tree

2.1.3 Heuristic rules

Heuristics-based approaches are the best-known approaches to NLP-based conceptual data modeling because heuristics, often guided by common sense, provides good but not necessarily optimal solutions to many difficult problems such as automated conceptual data modeling where precise algorithmic solutions are not available. All of the previous research made some use of heuristic rules, although some of them might be implicit. Some of the heuristic translation rules were general, while others were very specific and language-dependent (some of the research was in languages other than English, e.g. German, French, Spanish).

Chen [Che83] studied the corresponding patterns between English sentence structures and the basic elements of ERDs, and proposed eleven high level heuristic translation rules for translating English information requirements into ERDs, as shown in Figure 7. The basic constructs of English, such as noun, verb, adjective, adverb, gerund, and clause were found to have their counterparts in the ER model. These high level heuristic rules are fundamental and have been adopted and extended in some other research.

Omar *et al.* [OHM04] developed an extension of Chen's heuristic rules to assist the semi-automated generation of ERDs. The heuristic rules were classified into five categories:

RULE 1: A common noun (such as "person", "chair") in English corresponds to an entity type in an ER diagram.

RULE 2: A transitive verb in English corresponds to a relationship type in an ER diagram.

RULE 3: An adjective in English corresponds to an attribute of an entity in an ER diagram.

RULE 4: An adverb in English corresponds to an attribute of a relationship in an ER diagram.

RULE 5: If the sentence has the form: "There are … X in Y", we can convert it into the equivalent for "Y has … X".

RULE 6: If the English sentence has the form "The X of Y is Z" and if Z is a proper noun, we may treat X as a relationship between Y, Z. In this case, Y and Z represent entities.

RULE 7: If the English sentence has the form "The X of Y is Z" and Z is not a proper noun, we may treat X as an attribute of Y. In this case, Y represents an entity (or a group of entities), and Z represents a value.

RULE 8: The objects of algebraic or numeric operations can be considered as attributes.

RULE 9: A gerund in English corresponds to a relationship-converted entity type in ER diagrams.

RULE 10: A clause in English is a high-level entity type abstracted from a group of interconnected low-level entity and relationship types in ER diagrams.

RULE 11: A sentence in English corresponds to one or more entity types connected by a relationship type, in which each entity type can be decomposed (recursively) into low-level entity types interconnected by relationship types.

Figure 7: Eleven heuristic rules proposed by Chen [Che83]

Heuristics that determine entities:

HE2: A common noun may indicate an entity type.

HE3: A proper noun may indicate an entity.

HE7: If consecutive nouns are present, check the last noun. If it is not one of the words in set S, where S = {number, no, code, date, type, volume, birth, id, address, name}, most likely it is an entity type. Else it may indicate an attribute type.

Heuristics that exclude non-potential entity time candidates:

HEX: A noun such as "record", "database", "company", "system", "information" and "organization" may not be a suitable candidate for an entity type.

Heuristics that determine attributes:

HA8: If a noun is followed directly by another noun and the latter belongs to set S, where S = {number, no, code, data, type, volume, birth, id, address, name}, this may indicate that both words are an attribute. Else it is mostly to be an entity.

Heuristics that determine relationships:

HR4: A verb followed by a preposition such as "on", "in", "by" and "to" may indicate a relationship type.

Heuristics that determine cardinalities:

HC2: The adjective "many" or "any" may suggest a maximum cardinality.

HC3: A comparative adjective "more" followed by the preposition "than" and a cardinal number may indicate the degree of the cardinality between two entities.

Figure 8: Some of the heuristic rules proposed in [OHM04]

rules that determine entities, rules that exclude non-potential entity type candidates, rules that determine attributes, rules that determine relationships and rules that determine cardinalities. Figure 8 contains some of the heuristic rules. These rules were associated with weights reflecting the confidence levels based on both statistics and general common sense. For example, HE2 ("a common noun may indicate an entity type") was given a weight of 0.5, which means that 50% of the time this heuristic will produce the correct result (because not all nouns are entity types).

SP1:

[Ng_subject] (OWNER) [verbal form](ownership_subject)

[Ng_complement](OWNED)

RL1 :

IF meaning(clause(verbal form)) = ownership_subject

AND gram_structure(Ng_subject) = <article, noun_1>

AND gram_structure(Ng_complement) = <article, noun_2>

THEN case(noun_1) = OWNER

case(noun_2) = OWNED.

RL2 :

IF meaning(clause(verbal form)) = ownership_subject

AND gram_structure(Ng_subject) = <article, noun_1, predicate_1>

AND gram_structure(Ng_complement) = <article, noun_2>

AND gram_structure(predicate_1) = <preposition, article, noun_3>

THEN case(noun_1) = OWNER(verb)* and OWNED(predicate)

case(noun_2) = OWNED.

Figure 9: Some of the heuristic rules used in [RP92]

In contrast to Chen and Omar's explicit heuristic translation rules, the rules used by Rolland & Prox [RP92] were implicitly embedded in the underlying natural language understanding modules. Three distinct classes of rules: lexical and syntactic rules, linguistic rules and mapping rules were used. The system used pattern matching to unify each syntactic tree with one of the sentence patterns. The lexical rules determine the grammatical nature of each word of any clause of a sentence and classify the verb clauses into four classes: ownership, action, state, and emergence. The syntactic rules were used to verify that a sentence belongs to the authorized language (grammatically correct). Pattern recognition was based on both the class of the verb and the grammatical structure of the sentence through a set of linguistic rules. For example RL1 and RL2 are two examples of rules necessary for implementing the pattern rule SP1 in Figure 9. Mapping rules were used to build the semantic net. Each

	Argument 1	Argument 2	Relation
case1:	Entity	Entity	Yes
case2:	Entity	Entity	No
case3:	Attribute	Entity	No
case4:	Does not Exist	Entity	No
case5:	Entity	Attribute	No
case6:	Attribute	Attribute	No
case7:	Does not Exist	Attribute	No
case8:	Entity	Does not Exist	No
case9:	Attribute	Does not Exist	No
case10:	Does not Exist	Does not Exist	No

Figure 10: Binary rule cases [GSD99]

syntactic tree was mapped onto a set of nodes and arcs of the semantic net from cases and patterns determined by the linguistic rules.

Similar to Rolland & Prox's approach, Gomez *et al.* [GSD99] used two kinds of rules: specific rules and generic rules. The specific rules were defined for a verbal concept when its semantics indicated that an action specific to the concept must be performed by the ER generator. It included three categories: verbal concepts defining hierarchical relations, such as "a manager is an employee", verbal concepts that introduce attributes, such as "the source, the time, and the location of each document are stored", and verbal concepts that define key attributes, such as "Items are identified by item type". The generic rules were not associated with any particular verbal concept and their actions were based only on the arguments of the NL-relation and the current state of the database design. They were subdivided into unary rules, binary rules and n-ary rules. Unary rules were used for the definition of attributes. When an NL-relation had a single argument, three cases were considered. (a) The argument had already been defined as an ER entity; in this case, the program would ask the user to determine if there was another ER entity. If so, an ER relationship would be created. Otherwise, no action would be taken. (b) It had been defined as an attribute. In this case, the program would ask whether the argument was an attribute of an existing entity. If so, the argument would be defined as an attribute of the user specified entity. Otherwise, no action would be taken. (c) It did exist in the database model under construction. Then it would

be defined as an entity. Binary rules may define attributes, ER-entities or ER-relationships. Under this rules, 10 cases were taken into consideration for each possibility of the arguments as shown in Figure 10. N-ary rules were used for the definition of ER-relationships. Besides these rules, there were also some very complex rules used for normalization, for example,

> Rule 1: If the logical form of the current clause contains an actor and a theme, and the NP of the actor or theme has a nominalization in head position, modified only by an article, a quantifier or nothing (the NP has the form [article or quantifier or nothing + nominalization]), and the NL-relation referred by the nominalization already exists in the current database model as an ER-relationship, say relation r_i, then make the thematic role whose NP does not have the nominalization an argument in the ER-relationship r_i.

2.1.4 Semantic analysis

Semantic analysis that deals with meaning representation and manipulation is an important component for sophisticated and intelligent conceptual data modeling applications.

2.1.4.1 Semantic representation The most commonly used semantic representation languages in natural language processing are First Order Predicate Calculus (FOPC), semantic networks and frame-based representations [JM00]. This is also the case in NLP-based conceptual data modeling applications.

FOPC provides a sound computational basis for formal meaning representation of requirements specifications. The terms, predicates and connectives in FOPC acquire their meanings by virtue of their correspondence to objects, properties and relations in the external world being modeled. Gomez *et al.* [GSD99] represented the sentence "XWZ, a company, sells shoes to many customers" in FOPC as

$$\exists(x)\exists(y)(Shoe(x) \wedge Customer(y) \wedge Sell(XWZ, x, y)).$$

If the sentence were "all companies sell shoes to customers", then the FOPC would be

$$\forall(x)(Company(x) \Rightarrow \exists(y)\exists(z)(Shoe(y) \wedge Customer(z) \wedge Sell(x, y, z)).$$

The transformation from natural language to FOPC is a complicated process. Gomez *et al.* [GSD99] used special frame-like knowledge representation structures: object-structure and a-structure, to capture the relations and their arguments,

```
(args(XWZ)(shoe)(customer)),
(vc(sell),(actor(XWZ(q(constant)))),
(them(shoe(q(?)))),
(to-poss(customer(q(many)))).
```

It states that there are three arguments (*XWZ*, *shoe* and *customer*); the NL-relation is *sell*; *XWZ* (a unique constant) is the *actor*; *shoe* with unknown quantifier is the *theme* and *customer* with a *many* quantifier is the *to-poss* thematic role.

Similarly, Osborne & MacNish [OM96] explored the use of logical forms to represent the relations specified in natural language assertions. A logical form is an expression in FOPC, augmented with *event* variables and generalized quantifiers. For example, for the sentence "A brake is applied", the corresponding logical forms are:

$some(x1), entity(x1)$

$some(x2), sg(x2) \wedge brake(x2)$

$some(e1), apply(pres(e1), x1, x2)$

These logical forms represent "there is some entity $x1$, and some singular brake $x2$, such that at some present event $e1$, $x1$ applies the brake". Osborne & MacNish [OM96] did not explain how to derive such logical forms from natural language sentences.

Quantifiers are used in natural language, FOPC and ER modeling. Natural language has many quantifiers such as *all, some, many, few, etc.* The problem of handling natural language quantifier with standard FOPC has long been recognized. A set-based approach to approximate some of the problematic natural language quantifiers such as *many* and *most* in FOPC was proposed in [All95]. For ER modeling, there are three basic forms of cardinality (1:1, 1:N, M:N). It is more like FOPC than natural language in terms of quantifiers. However, FOPC is not ideal as an underlying meaning representation of ER formalism. As shown in the example above [GSD99], the quantifier *many* for entity type "customer" was lost in the final FOPC representation.

Semantic networks are also frequently used for meaning representation. Network notations and FOPC are both capable of expressing almost equivalent information. The major advantage of network representation is the ability to show direct connections, while FOPC must rely on repeated occurrences of variables or names to show the same connections

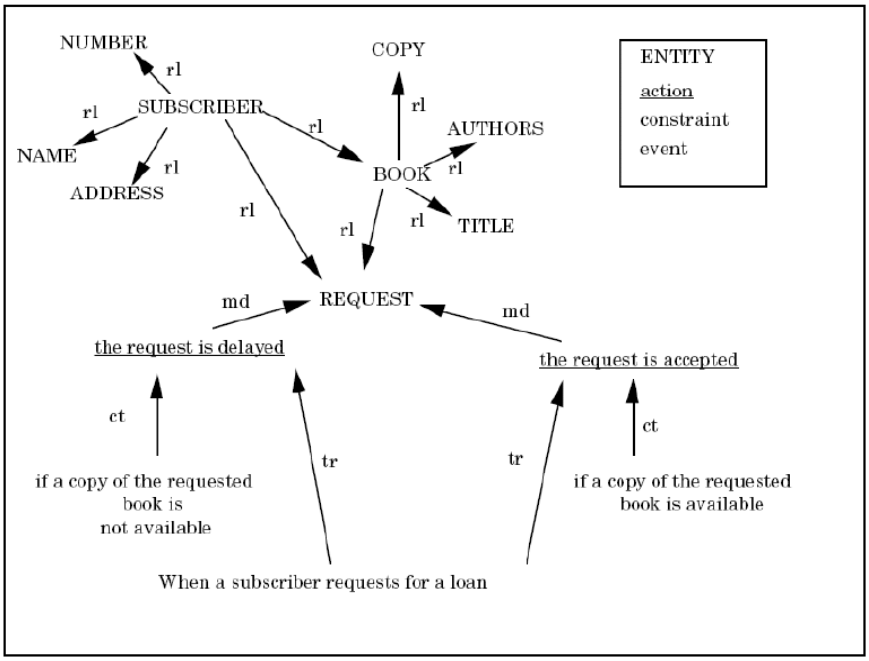


Figure 11: A semantic net [RP92]

[Sow91]. Variants of semantic networks exist and are used to represent word meanings, proposition, events and spatial relationships. Rolland & Prox [RP92] used a set of rules to map each syntactic tree onto a set of nodes and arcs in a semantic net as illustrated in Figure 11. The system was based on the REMORA methodology [Rol88] which identifies four basic concepts: objects, events, actions and constraints. These four concepts correspond to four types of nodes in the semantic net. The semantic net used here also includes five types of arcs: relationship arc, modification arc, trigger arc, constraint arc and event state change arc (not shown in the figure).

Mich [Mic96] utilized SemNet, a type of semantic network to hold knowledge that can be accessed, modified and expanded in the processing of object oriented software requirements. SemNet contains more than 100,000 connected nodes, which enable the underlying natural language understanding (NLU) subsystem to go beyond superficial linguistic structures to deep structures close to more generic cognitive forms. Concepts are represented as two types of nodes in SemNet: entity nodes and event nodes. The event nodes have a frame-like

structure representing the various components of the events, such as subject, action and object. Simple relations between concepts are represented as arcs.

2.1.4.2 Cases and thematic hierarchy Fillmore [Fil68] introduced case grammar to analyze sentences by a set of cases such as AGENTIVE and INSTRUMENTAL. He also formulated the basic idea that which argument of the verb will become the subject of a sentence [JM00]. This has motivated research on the mapping between conceptual structure and grammatical function. The following is a typical case priority ordering.[2]

$$\text{AGENT} \succ \text{INSTRUMENTS} \succ \text{THEME}\ [\text{JM00}]$$

It states that if case structure of a sentence includes an AGENT, then the AGENT will be realized as the subject; if the thematic description does not include an AGENT, then the INSTRUMENT will be realized as the subject. And if neither AGENT nor INSTRUMENT is present then the THEME will be realized as the subject.

Rolland *et al.* [RP92, RBA98, RP00] extended Fillmores case theory and applied case not only to words but also to clauses. The case approach was first applied to subordinate clauses with regards to the main verb, and then applied to the subordinate clauses. The system used the following cases ⟨OWNER, OWNED, ACTOR, TARGET, CONSTRAINED, CONSTRAINT, LOCALIZATION, ACTION, OBJECT⟩. For example, in the sentence "A subscriber is described by a name, an address and a number", "subscriber" is associated to the OWNER case and "name", "address" and "number" are associated to the OWNED case. A case grammar was also used by Fougeres [Fou99].

Gomez *et al.* [GSD99] utilized thematic roles which are also closely related to case theory. A semantic interpreter was used to identify thematic roles. For example, in the sentence "A company sells books to customers", "company" is identified as the *actor*; "sells" is identified as the *transfer-of-possession* and "books" is identified as the *theme*. This thematic role information is used to address partial prepositional phrase ambiguity. In this case, the prepositional phrase "to customers" is attached to the verb and stands for the thematic role *to-poss*. The ER generator in the system utilized these thematic roles to build ER formalisms. The tricky point here is that if the thematic roles were identified incorrectly, then the ER

[2]Different case priority orderings exist in the literature.

generator would generate the wrong ER formalism. For instance, in a similar sentence, "A company sells books about travel", if the semantic interpreter attaches the prepositional phrase "about travel" to "sells", but not "books", then the ER generator will produce three ER-entities: "company", "book" and "travel".

2.1.5 Ambiguity and disambiguation

Ambiguity is a pervasive phenomenon in natural language. According to the Oxford English Dictionary, each of the 500 most often used words in English has an average of 23 different meanings. The word "round", for instance, has 70 distinct meanings [Gra]. In fact, this is only the lexical ambiguity. There are also syntactic and semantic ambiguities. Ambiguity poses great challenges for applications of constructing unambiguous and consistent target formalisms.

2.1.5.1 Word sense disambiguation The approaches to word sense disambiguation can be categorized as symbolic methods, knowledge-based methods and statistical corpus-based methods [IV98].

The symbolic methods refer to some of the AI work from 1960's to 1980's. Quillian [Qui67] built a kind of semantic network that linked words and concepts with various semantic relations. Disambiguation was accomplished when only one concept node associated with a given input word was activated by a path finding program. Hayes [Hay78] used a combination of a semantic network and a version of case frame for word sense disambiguation. Senses of nouns were presented as nodes and senses of verbs were represented as links in the semantic network. The network also incorporated *is-a* relation as in semantic networks and *part-of* relations as in frames. It worked fairly well on disambiguation of homonyms. Andriaens & Small [AS88] utilized a discrimination net in the *Word Expert Parser* system which included multiple word expert subsystems. Unique sense was achieved by traversal of the discrimination net.

The symbolic methods were usually based on a particular linguistic theory with a limited hand-crafted knowledge base. With the availability of large-scale general knowledge sources

in the late 1980's, word sense disambiguation work shifted towards empirical methods utilizing these resources. Cowie *et al.* [CGG92] utilized the box codes (abstract, animate, human, etc.) from the electronic version of the Longman Dictionary of Contemporary English and reported results of 47% for sense distinction and 72% for homographs. Yarowsky [Yar92] worked on disambiguating new occurrences of polysemous words by deriving words classes from Roget's International Thesaurus. Bayes' rule was applied to determine the most likely classes of a polysemous word. He reported fairly good results on noun disambiguation. Sussna [Sus93] worked on noun sense disambiguation by using a semantic distance metric computed from the relation links in WordNet. He explored not only the *is-a* relation but other relation links in WordNet.

Statistical methods for word sense disambiguation revived in the 1990's with the availability of several large-scale linguistic corpora. Hearst [Hea91] trained his algorithm on a manually sense-tagged corpus, and then he used statistical information obtained in the training phase to disambiguate other occurrences. Brown *et al.* [BPPM91] took advantage of the statistical information in a bilingual corpus to assign word sense by the context in which the word appeared based on the assumption that different senses of a given word often translate differently in another language. Yarowsky [Yar95] proposed an unsupervised learning algorithm trained on unannotated English text for sense disambiguation. The algorithm is based on the assumption that the sense of a target word is highly consistent within any given document and nearby words provide strong and consistent clues to the sense of the target word. A 96% accuracy was reported.

Gomez *et al.* [GSD99] used WordNet as a knowledge source for word sense disambiguation in database conceptual data modeling. An interface was used to access WordNet and display the ontological categories for a given word. When ambiguity happens, the system asks the user to choose the proper ontological category in the current context. The authors didn't automated the disambiguation process by consulting some kind of knowledge base such as Cyc [LG90] or WordNet [Chr98].

Surprisingly, word sense disambiguation in NLP-based conceptual data modeling is arguable. On the one hand, word sense is a basic component of semantic analysis, and it is important for constructing correct and consistent formal representations. On the other

hand, for heuristic rules based on word classes, as long as the POS of a word is identified correctly, the different senses may be not significant for the construction of certain target formalisms. For example, "bank" can be either "a financial institute" or "slopping land". In ERD sense, as long as it is identified as a noun and treated as an entity correctly, the sense chosen is not important since it does not affect the final conceptual data model.

2.1.5.2 PP attachment and conjunction disambiguation Pre-positional phrase attachment ambiguities account for a good deal of the ambiguities in requirements specification documents, as they do in English in general. When a sentence is in conjunction form (mainly including one or more "and"), it becomes even more difficult for prepositional phrase attachment disambiguation. Approaches from detailed knowledge-based methods to shallow quasi-statistical approaches based on empirical evidence found in large corpora have been tried in previous research. Corpus-based prepositional phrase attachment disambiguation methods [BR94, RRR94, CB95, ZDJV97] were reported to achieve up to 84.5% accuracy on the Penn Tree Bank corpora. A better result was reported in [SN97] with 88% accuracy using the WSJ text in conjunction with WordNet.

Gomez *et al.* [GSD99] utilized special knowledge structures, which include an object-structure (essentially a frame-like structure for nouns) and an a-structure (essentially a frame-like structure for verbs), to resolve prepositional phrase attachment ambiguity for conceptual data modeling. The algorithm was detailed in [GSH97]. The basic idea is that if a verb strongly claims a preposition by certain rules, the preposition will be attached to the verb; if a verb weakly claims a preposition, the preposition will be attached to both the verb and the NP temporarily. The ambiguity is resolved later by other attachment evidence.

2.1.5.3 Reference resolution One often used approach for reference resolution is searching for NPs in the current and preceding sentences, then applying a set of resolution constraints and preferences, such as gender and number agreement, c-command constraints [IS89], semantic consistency, syntactic parallelism, salience, proximity, *etc.* [Mit99].

Another approach is to take into account the effects of discourse structure. Grosz [Gro77] studied definite reference resolution in task-oriented dialogues. Grosz's approach was to find

ways of determining and representing the explicit and implicit focus of attention of a discourse as a means for constraining reference resolution. Grosz's work relied heavily on task-oriented discourse structure and demonstrated the role of discourse structure in reference resolution.

Sidner [Sid78] developed a similar focus of attention approach for reference resolution in the PAL personal assistant program. Hirst [Hir81] reviewed several other discourse-oriented anaphora resolution approaches. Knowledge-rich methods [LL94], knowledge-poor methods [Mit98] and statistical methods [DI90] have been reported for reference resolution in the literature.

Lee [Lee03] claimed that pronouns in requirements documents have relatively small scope, usually referencing the words in the same sentence or the previous sentence, and it is rare that the pronouns "it" and "that" are used to refer to a word that is not present in the requirements documents. So he used common heuristics that incorporate the two approaches discussed above with both recency constraints (the recent word has a higher priority than less recent ones) and the discourse focus (the coreferred word has a higher priority than words that are not coreferred) to dereference pronoun. Lee & Byrant [LB02b] also reported the utilization of WordNet for anaphora reference resolution, but no detailed information was provided.

Gomez *et al.* [GSD99] utilized a program called SNOWY [GSH97] to store input concepts in its long term memory in a hierarchical fashion for reference resolution. The current version of the database design and the SNOWY program were combined to resolve anaphoric references in the current input sentence while incrementally developing the database design. For instance, for the following input,

> Each person keeps a record of documents of interests. The source and the time of the documents are stored.

the concepts "document", "documents-of-interest" and the relation "document-of-interest is-a document" were stored in SNOWY from the first sentence. Meanwhile, the E-R generator built a data model including the ER-entity "person" with attribute "document of interest". When the second sentence was read, the ER generator realized that "the document" was not in the current data model and would access the SNOWY's hierarchy under "document" to resolve the reference (in this case the "document-of-interest"). The system did not use a

generalized knowledge base such as WordNet but required users' input for problems it could not resolve itself.

A quite different approach was adopted by Pulman *et al.* [PAC+93] for reference resolution in a natural language database interface. The reference resolution mainly focused on compound nominals, such as "London buses" and "DTI project". For instance, "London buses" may mean "the buses in London" or "the buses to London" in different contexts. In order to resolve such references, sentences were analyzed into quasi-logical forms [ACRG91] to represent the basic linguistically determined meaning of the sentences. The quasi-logical forms contain constructs for compound nominals, pronouns, ellipsis, *etc.*, which enable reference resolution rules to suggest a contextually plausible resolution of the intended meaning. The resolution rules were in the format,

$$A \leftrightarrow B \quad if \quad C_1, \cdots, C_n$$

Stating that A is equivalent to B if the conditions $C_1, \cdots, C_n$ hold. The conditions may refer to aspects of the context, or to the domain model. However, no detailed information and examples of these resolution rules were given except the total number of rules that were implemented in the system. The problem for such an approach is that a large number of resolution rules are needed even for a small application.

2.1.5.4 Scope ambiguities Scope ambiguities are usually caused by quantifier operators. For instance, a sentence like "Everybody loves someone" can mean either that for each person there is someone that he loves or that there is someone everybody loves (although unlikely). These two meanings corresponds two logical forms: $(\forall x)(\exists y)Loves(x, y)$ and $(\exists y)(\forall x)Loves(x, y)$.

Quantifier words such as "every", "each", "any", "all", "many", "some", "several", "a", "an", *etc.*, and negation words, such as "no", "not" and "never", are very common in requirements specification documents. The author has done a comparison of the frequency of common scope words in a small requirements documents corpus to the frequency in an English news text corpus TDT2 [SM00] (see Appendix B). It shows that the percentage of scope words is much higher in requirements documents than in news text[3]. However, this

[3]It may also be higher than in general English

problem has not been addressed in NLP-based conceptual data modeling. The reason may be that, on the one hand, scope ambiguities usually involve subtle semantics of the sentences, so it is hard to resolve; and on the other hand, word morph (single/plural) and cardinal numbers can be used to extract the necessary relationship cardinality requirements.

2.1.6 Domain and background knowledge

Requirements documents written in natural languages are intended for human readers. Some background and common sense knowledge is usually assumed. For instance, if the requirements state "the user inputs the 4-digit PIN number by pressing the buttons", it assumes the reader knows the fact that the set of buttons is a component of the ATM machine, and therefore it is not explicitly mentioned in the requirements document [LB02c]. Most of the time, such background and common sense knowledge that describes the relationship between components and other constraints in requirements documents is too implicit to be extracted from the original documents.

Lee & Bryant [LB02b, Lee03] specified the domain knowledge of a system in DAML (DARPA Markup Language), a frame-based language with an expressive semantics to facilitate the development of the Semantic Web. Figure 12 illustrates an example of the domain knowledge representation in the bank domain, which states the fact that "ID, PIN, and balance are three components of an account".

The SemNet utilized by Mich [Mic96] is an example of exploring a large-scale general background knowledge source for conceptual data modeling. The SemNet contains more than 100,000 connected nodes after it was merged with WordNet [Chr98]. The SemNet was used in the context independent phase for domain analysis and disambiguation purpose. The requirements documents were extended by adding more SemNet nodes in the modeling process. However, the details of their approach were not disclosed in the paper.

For automated conceptual data modeling, large scale domain knowledge bases are desired [Sto02]. However, even for a very specific domain, to build an exhaustive, detail-oriented knowledge base (domain ontology) is labor intensive and time-consuming. Buchholz *et al.* [BCD+95] carried out a series of interviews with librarians as well as library users to obtain

```
<daml:Class rdf:ID="Account">
  <daml:disjointUnionOf
    rdf:parseType="daml:collection">
    <daml:Class rdf:ID="ID"/>
    <daml:Class rdf:ID="PIN"/>
    <daml:Class rdf:ID="Balance"/>
  </daml:disjointUnionOf>
</daml:Class>
```

Figure 12: A domain knowledge representation [LB02b]

a lexicon of 12,000 items in the library domain. The Cyc project [LG90] has taken more than 20 years several dozen human cyclists to build a large scale common sense knowledge base.

2.1.7 Target formalisms

Target formalisms refer to the explicit and formal representations of conceptual data models. They can be either diagrammatic or in other forms. ERD and UML class diagrams are two of the often used diagrammatic formalisms in automated conceptual data modeling. ERD is commonly used in database design applications, and UML class diagrams are widely used in software engineering [BMO01].

Figure 13 illustrates an ERD model of the natural language database requirements from the previous section. Chen [Che83] detailed how to transform a natural language requirements specification to such an ERD model with 11 heuristic rules, which were refined and extended by several other researchers [TB93, GSD99, OHM04, DM06] (see section 2.1.3 heuristic rules).

For object-oriented analysis, UML class diagrams are the preferred target formalism for conceptual data modeling, especially in software design. Many software design tools (section 1.1) offer functionalities to draw UML diagrams and transform UML to various source

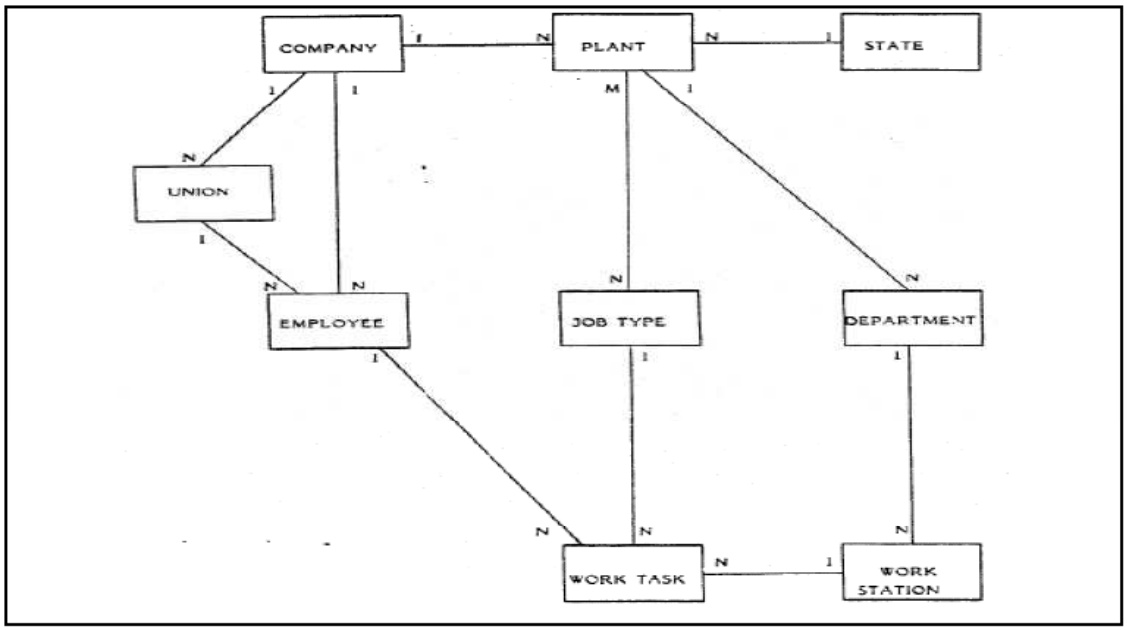


Figure 13: Output formalism ERD [Che83]

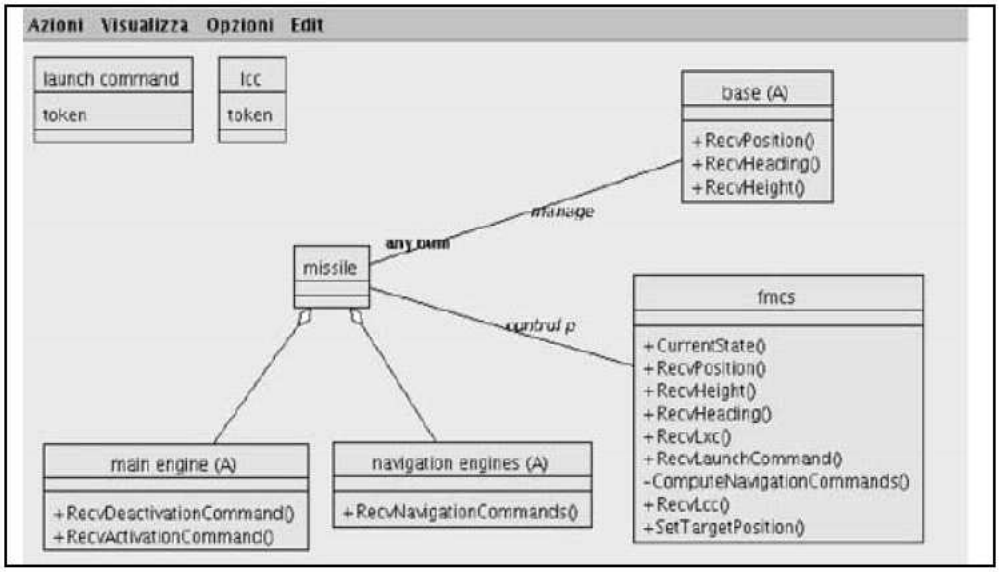


Figure 14: Output formalism UML [AG06]

codes, but not the transformation from natural language requirements to UML formalisms. The transformation from natural language to UML is similar to the translation from natural language to ERD but is more difficult, because the scope of UML is broad, especially the UML 2.0 specification. It includes many representations that are redundant and infrequently used. Many of the standard elements have vague semantics and are inconsistently named. UML is specified by a combination of its own abstract syntax and English, and in some cases, they are inconsistent with each other [Kob99]. Among the diagrams that are specified in the UML, class diagrams, which have consistent semantics, are frequently used. Figure 14 illustrates a UML class diagram generated from a natural language description about a fictitious missile control system [AG06]. Mich [Mic96] and Harmain & Gaizauskas [HG03] also adopted similar object-oriented formalisms.

Other formalisms were also utilized in conceptual data modeling. TELL [SHE89] was a specific formal specification language for concurrent systems. Its formal semantics are given by a temporal logic, both in textual expressions and graphic expressions. ACTL [FGR+94] was an Action-based Computation Tree Logic (as shown in Figure 15) used for the formalization of behavioral requirements in the design of reactive system. Conceptual graphs [Sow84] were adopted by Fougeres & Alain [Fou99] as a formalism for the representation of semantic knowledge.

```
Sentence: After a coin has been inserted, it is possible to
          have tea.
Parsed in: 8.150 seconds.
ACTL formula: [coin] E[true {false} U (Hristea) true]
Sentence: After a coin has been inserted it is possible to have
          a soup until another coin is inserted.
Parsed in: 16.783 seconds.
ACTL formula: [coin] A[true {~soup} U {coin} true]
```

Figure 15: An output of the NL2ACTL [FGR+94]

2.2 EVALUATION METHODS

Both qualitative and quantitative approaches were used for evaluating the system performance in previous research.

2.2.1 Qualitative evaluation

2.2.1.1 Case study Case study is one of the often used empirical qualitative methods to demonstrate the usage and the performance of application systems. Case study usually involves an in-depth examination of a single instance or example. Chen [Che83] provided a case study to demonstrate the applicability of the eleven heuristic rules he proposed. The analysis process was based on human reading of the text without any automation. For instance, he said the sentence "The company has 50 plants located in 40 states and approximately 100,000 employees" could be decomposed into three sub-sentences (clauses): (1) the company has 50 plants; (2) the 50 plants are located in 40 states; (3) the company has approximately 100,00 employees. Then his rules could extract the possible entities and relationships. However, considering the coordinating conjunctions and prepositional phrase attachment ambiguities in this sentence, many of the state of the art parsers have difficulty parsing this complex sentence into the constituent sentences [DM06]. After analyzing each sentence of the example text, Chen compared his ERD with Teorey's [TF82] and pointed out the pros and cons of applying his heuristic rules for ERD building.

Lee & Bryant [LB02b] demonstrated the capability and performance of their automated software requirements transformation system on the Computer Assisted Resuscitation Algorithm Infusion Pump Control System requirements [WRA01]. The outputs of each step, from natural language requirements to XML, to hierarchical knowledge base, to Two Level Grammar, and to the final formalism VDM++ were examined and explained. Similar approaches were adopted in [FGR+94, NR95, Mic96, OM96, AG06].

2.2.1.2 Requirements validation As illustrated in Figure 1, requirements validation is an important phase in requirements engineering that helps in checking the correctness

and consistency of software systems. Customers' participation is desirable for requirements validation. However, the target formalisms generated are aimed at information analysts, not at customers. Some previous systems incorporated natural language generating and paraphrasing capabilities for the purpose of requirements validation.

Rolland & Prox [RP92] utilized three classes of rules: extraction rules, transformation rules and linearization rules, to generate natural texts from conceptual schema for the purpose of requirements validation. Extraction rules were used to cluster nodes and arcs related to either entities or event types in the semantic net (see section 2.1.4) to construct deep structures. Transformation rules were used to map deep structures into surface structures following Chomsky's theory. Linearization rules, including rules for conjugating the verb and selecting correct articles, were used to rewrite a surface structure into readable sentences. The aim of this process was to generate readable language as close as possible to the original requirements documents. So the customers without any expertise in conceptual data modeling can compare and verify the correctness and consistency of the formal conceptual schema.

Burg & van de Riet [BvdR96] claimed that the transformation of formal specifications into informal specification could also be helpful to the analysts themselves, who can see the specification from a new perspective. He utilized a tool, CPL2NL, to automatically generate readable paragraphs from conceptual prototyping language specifications (Figure 16). Although the language generated was not as readable as the input text (Figure 4), it was usable for the validation purpose.

```
[an,user,is,permitted to,borrow,one or more,books,from,a,library]
[if,an,user,borrowed,one or more,books,from,a,library,at,time,T,
then,an,user,will have to,return,one or more,books,to,a,library,
at,time,T2, where,T2,is less or equal,T + 3 * week]
[an,user,borrowed,a,book,at,a,time,T1,where,T1,is,yesterday]
```

Figure 16: A paraphrasing example from CPL2NL [BvdR96]

2.2.2 Quantitative evaluation

Methods and measures originally developed for evaluating information retrieval systems, i.e., *recall* and *precision* [vR79] were adopted to provide quantitative evaluation of the performance of the systems in some of the previous research.

Gomez *et al.* [GSD99] were the first to provide quantitative evaluation of conceptual data modeling systems. However they did not state the performance in terms of recall and precision explicitly. Harmain & Gaizauskas [HG03] were the first to introduce precision and recall as the measures for evaluation of conceptual data modeling systems. Omar *et al.* [OHM04] extended Harmains definition and proposed more measures.

Recall is the percentage of the total relevant information that is actually found in a particular situation. In conceptual data modeling, it is the number of correct items (such as entities, objects, relations, attributes) returned by the system compared with those produced by human analysts or answer keys. Harmain & Gaizauskas [HG03] proposed

$$Recall = \frac{N_{correct}}{N_{key}} \quad \text{[HG03]} \tag{2.1}$$

$N_{correct}$ refers to the number of correct responses made by the system. N_{key} is the number of items in the answer keys. Omar *et al.* [OHM04] redefined recall with different terms but with same semantics,

$$Recall = \frac{N_{correct}}{N_{correct} + N_{missing}} \quad \text{[OHM04]} \tag{2.2}$$

$N_{correct}$ is the same as above. $N_{missing}$ is the number of correct items that are not extracted by the system.

Precision is the percentage of all extracted information that is actually correct in a particular situation. It reflects the accuracy of the system in obtaining the correct results. Harmain & Gaizauskas [HG03] proposed

$$Precision = \frac{N_{correct}}{N_{correct} + N_{incorrect}} \quad \text{[HG03]} \tag{2.3}$$

$N_{correct}$ is the same as above. $N_{incorrect}$ refers to the incorrect responses made by the system. Omar *et al.* [OHM04] used a slightly different measure by taking more factors into account.

$$Precision = \frac{N_{correct}}{N_{correct} + N_{incorrect} + N_{ask} + N_{missing} + N_{overgenerate}} \quad \text{[OHM04]} \tag{2.4}$$

$N_{correct}$, $N_{incorrect}$ are the same as above. N_{ask} is the number of user assistance acts in respondence to the system requests. $N_{overgenerate}$ is the number of correct items that are generated by the system but are not in the answer keys. It is questionable why Omar *et al.* also put $N_{missing}$ in the denominator.

Besides *recall* and *precision*, new measures that are suitable for the evaluation of conceptual data modeling systems have been defined. There is usually no single gold standard model, as different people may produce different models or similar models with synonyms for a given requirements specification. Harmain & Gaizauskas [HG03] used Over-specification and Omar *et al.* [OHM04] used Overgenerated to measure how much extra correct information is in the system response but is not found in the answer keys.

$$Over\text{-}specification = \frac{N_{extra}}{N_{key}} \quad \text{[HG03]} \tag{2.5}$$

$$Overgenerated = \frac{N_{overgenerated}}{N_{correct} + N_{missing}} \quad \text{[OHM04]} \tag{2.6}$$

Omar *et al.* [OHM04] also proposed several other measures. *Undergenerated* is the percentage of total relevant information that is not extracted by the system.

$$Undergenerated = \frac{N_{missing}}{N_{correct} + N_{missing}} \quad \text{[OHM04]} \tag{2.7}$$

Ask_user is the ratio of user assistance requests generated by the system to the total relevant information. This measure evaluates the importance and the frequency of user interventions to a system.

$$Ask\_user = \frac{N_{ask}}{N_{correct} + N_{missing}} \quad \text{[OHM04]} \tag{2.8}$$

Unattached is the ratio of ER elements that are correctly identified by the system but are not attached to their corresponding items to total correct items.

$$Unattached = \frac{N_{unattached}}{N_{correct} + N_{missing}} \quad \text{[OHM04]} \tag{2.9}$$

Similarly, wrongly_attached is the ratio of ER elements that are correctly identified by the system but are wrongly attached to other items to the total correct items.

$$Wrong\_attached = \frac{N_{wrong\_attached}}{N_{correct} + N_{missing}} \quad \text{[OHM04]} \tag{2.10}$$

Table 1: A comparison of the performance of previous systems

Systems	**Evaluation Results**	
	Recall	Precision
ER Generator [GSD99]	75%	-
CM-Builder [HG03]	73%	66%
ER-Converter [OHM04]	95%	82%
ER-Builder [DM06]	93%	90%

Note: similar but different datasets are used for the evaluation of the above systems.

A comparison of the recall and precision of several of the previous implemented systems is presented in Table 1. Gomez *et al.* [GSD99] stated that the ER Generator was able to identify all the relevant ER relationships and entities in 75% of the cases that were interactively input by the users. However, the program overgenerated or undergenerated ER entities and relationships in approximately 50% of cases in a corpus of 45 problems collected from database text books and other resources. The CM-Builder [HG03] was evaluated on a corpus of five case studies extracted from textbooks. These case studies range from 100 to 300 words, and the average sentence length is 17 words. The overall performance of the system was 73% recall and 66% precision, with an over-specification of 63%. The ER-Converter [OHM04] was tested on a test dataset which consisted of 30 database problems gathered from database books and past exam papers. The problems ranged from 50 to 100 words in

size. It achieved a high average recall of 95% and 82% precision. The ER-Builder [DM06] was evaluated on a corpus consisting of 113 sentences taken from 20 database specification documents. The average sentence length is approximately 11 words. The recall was about 93% and precision was about 90%.

Harmain & Gaizauskas [HG03] claimed that some of the machine translation, information retrieval and extraction systems with much lower precision and recall have found commercial applications, so there should be possible practical applications of automated conceptual data modeling. However, the requirements for precision and recall in different domains and applications are different, and such requirements are usually high in automated conceptual data modeling. Furthermore, all of the above evaluations were based on small sets of data. A relatively large and publicly available corpus of requirements documents is needed for better quantitative evaluation of automated conceptual data modeling systems.

3.0 RESEARCH APPROACH

The general research objective of interest is exploring feasible methods to transform semi-restricted natural language documents into a relatively formal representation. Although a specific type of semi-restricted natural language documents, database requirements specifications, and a specific formalism, Entity Relationship Diagram(ERD), was chosen in this research, it is expected that the general approach can be applied into many other similar fields such as medical records extraction and traffic accident records extraction that require transforming semi-restricted natural language documents to formal representations.

3.1 OVERVIEW

The transformation of general free texts to unrestricted formal knowledge representations such as FOPC is not yet feasible. However, it is increasingly feasible to translate semi-restricted NL documents such as database requirements specifications into semi-restricted formalisms such as ERD. Currently, parsing technologies are relatively mature and many of the state of the art parsers are readily available, so this research focuses on analyzing the sublanguage characteristics of a specific example of semi-restricted documents, extracting specific information based on parsing results and exploring the problem of disambiguation as it appears in the domain of ER modeling.

In database specifications, sentences can be treated as independent units that specify either an entity to entity relation or an attribute attachment to an entity. It has been found that some of the sentence patterns, especially the long attribute enumerative sentence structures are difficult for state of the art parsers. It is possible to develop specific

pattern matching approaches for these cases in this research, rather than working on these problematic parsing results.

As in any natural language understanding application, an NLP parser is needed for syntactic (grammatical) analysis of the input sentences. Theoretically, any NLP parsers will work to some extent, but the overall performance of the automated conceptual data modeling system depends heavily on the parsing results. Also many other practical factors such as availability, format of parsing results, were involved when choosing NLP parsers. Parsing results from two of them: the Link Parser [ST91] and the Stanford Parser [KM02], as reviewed in section 2.1.2, were investigated. Both of the parsers provide high level dependency-related grammatical relations with some trade-offs. For instance, the parsing results from the Link Parser are more precise for well-formed sentences than those from the Stanford Parser, while the Stanford Parser is more robust on handling ungrammatical sentences than the Link Parser.

Based on the linkages from the Link Parser or the Typed Dependencies from the Stanford Parser, various heuristic rules used to extract entity relationship tuples were proposed. Because of the complexity of natural language and the imprecision of the underling parsers, inappropriate tuples will inevitably be generated by the heuristic syntactic rules. So the elements extracted from those heuristic rules need to be further processed, especially those involving special structures. The original requirements text provides the most reliable source to resolve some of the ambiguities in the extracting results, however the requirements text itself may be informal and ambiguous too. Disambiguation based on WordNet, Web corpus and other large scale knowledge resources were investigated to filter out inappropriate elements.

In order to graphically represent the database requirements in ERD, the extracted entity relationship information was translated into DOT language [GN00] which can be rendered into various graphic formats. The open source graphic visualization tool packages Graphviz and Grappa [EGK+04] were used to generate the ERD and provide an interactive interface.

3.2 DATABASE REQUIREMENTS AS A SUBLANGUAGE

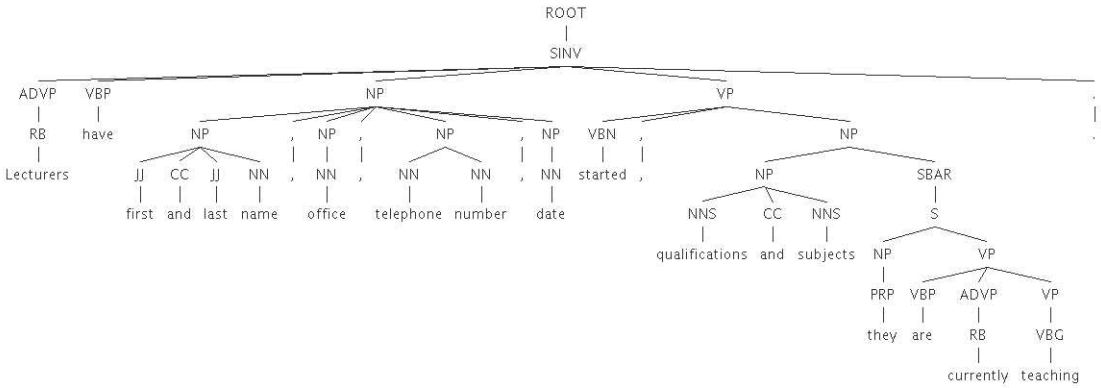
In terms of general requirements specifications, it seems there is no obvious vocabulary or domain restriction without specifying a particular domain such as financing or manufacturing. However, the language may still be treated as a kind of sublanguage because most of these documents exhibit some special regularities and restrictions on sentence structures. These include:

- A restricted set of sentential patterns. Usually only declarative sentence structures are used in requirements specification documents. Imperative, yes-no-question and wh-non-subject-question sentence structures seldom appear in requirements specification documents.
- Exclamation, interjection, weird words, slang, idioms, peculiar grammatical structures and incomplete sentences are unlikely to appear in this kind of documents.
- Relatively simple temporal expressions are used in the documents, e.g. there is a predominant use of the simple present tense.
- Enumeration of attributes, as a form of conjunction, is common.

In order to extract entity relationship from the database requirements specifications, correctly parsing the input is a necessary step. Some of the sublanguage characteristics such as restricted set of sentential patterns are helpful in reducing the complexity of the problem, while others such as the enumeration of attributes are less common in general natural language, and hence pose more challenges to the parsing tasks. From the analysis of a small database requirements corpus, it was found that even the most relevant and accurate parser for this purpose has difficulty parsing the long attribute enumerative sentences such as "lecturers have first and last name, office, telephone number, date started, qualifications and subjects they are currently teaching" which are often used to describe the attributes of an entity or a relation in database requirements. For instance, the Link Parser can not parse the above sentence while the Stanford Parser provides wrong results: the word "started" is labeled as the main verb which messes up the whole sentence, as shown in Figure 17.

Sentence:
Lecturers have first and last name, office, telephone number, date started, qualifications and subjects they are currently teaching.

Parse tree:




```
Typed Dependencies:

advmod(started-14, lecturers-1)          aux(started-14, have-2)
amod(name-6, first-3)                    cc(first-3, and-4)
conj(first-3, last-5)                    dep(started-14, name-6)
appos(name-6, office-8)                  nn(number-11, telephone-10)
appos(name-6, number-11)                 appos(name-6, date-13)
dobj(started-14, qualifications-16)      cc(qualifications-16, and-17)
conj(qualifications-16, subjects-18)     nsubj(teaching-22, they-19)
aux(teaching-22, are-20)                 advmod(teaching-22, currently-21)
rcmod(qualifications-16, teaching-22)
```

Figure 17: An example parse tree of a long attribute sentence

When the sentence structures are examined, it is not hard to find some patterns that can be used to extract the attribute list even without parsing the sentence. So instead of using the parsing results from these long enumerative sentences, a pattern matching approach is more appropriate to extract the desired elements from this kind of sentence by taking advantage of the sentence structure regularity. In addition, compared to the parsing results, the tagging results are usually more correct. The POS of the words from the tagging results plus the syntax of the sentences can be used in the patterns to extract the attribute information. Moreover, a small set of typically used attributes in database requirements specifications, such as "name, id, address, phone number" can cover a significant part of the

cases although there is no closed lexicon that defines the scope of the vocabulary that can be used to express entity attributes. The regular expression-based pattern matching approach is detailed in Section 4.1.3.

3.3 DEPENDENCY RELATED PARSING

Recently there has been an increasing use of dependency parsing for various NLP tasks, from machine translation to question answering [dMMM06] because of the high level grammatical relations offered between individual words. It seems that using heuristic rules based on high level information such as grammatical relations from full level dependency parsing is very promising for formalism extraction in automated conceptual data modeling. Two of the dependency related parsers were studied.

3.3.1 Link Parser

Link Parser[1] is based on Link Grammar, an original theory of English syntax [ST91]. It has a lexicon of about 60,000 word forms and covers a wide variety of syntactic constructions. The most attractive feature of Link Parser is the high level grammatical relations offered from the parsing result links. For example, Figure 18 illustrates the link graph of the sentence "the company has 50 plants located in 40 states".

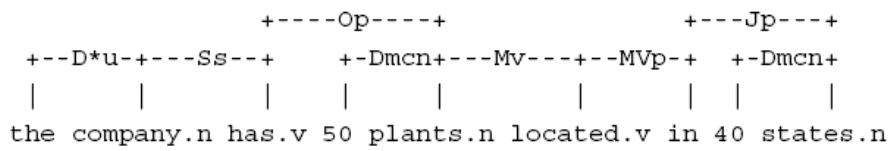


Figure 18: A Link graph generated by the Link Parser [ST91]

[1]There is no consensus on whether Link parser belongs to dependency parsers [Sch98].

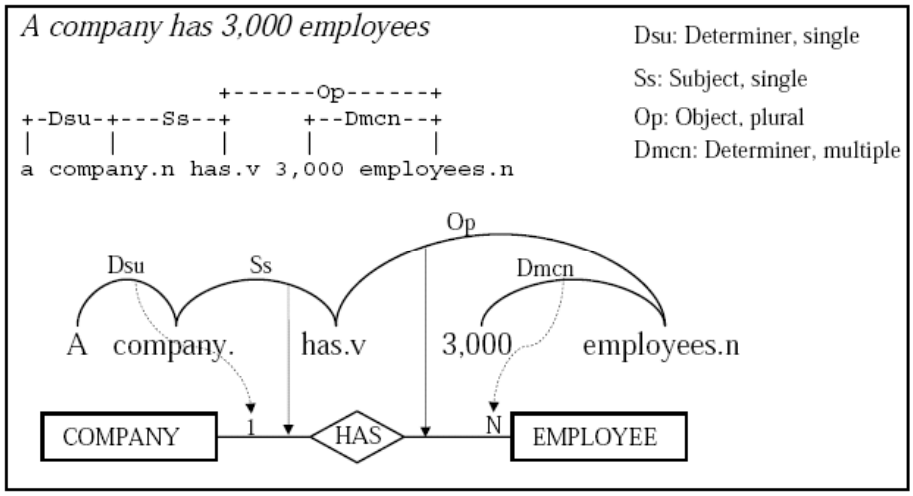


Figure 19: Links vs. ERD elements [DM06]

A link graph is constructed from compatible links connecting pairs of words. Each word has a set of left and right pointing connectors in the lexicon. The link graph is built by searching over the compatible links space. If there exists a complete linkage that covers all the words in a sentence (except conjunctions), then a valid parse is produced. The links are organized from left to right directly from words in a link graph which is different from the hierarchically organized tag sets in constituent trees where constituents are composed of sub-constituents.

Du & Metzler [DM06] argued that the reason for choosing the Link Parser over similarly powerful parsers was that there are relatively straightforward correspondences between the links and the components of entity relationship diagrams, as illustrated in Figure 19. The output structure of the Link Parser offers not only POS information, but also some deep sentence role (grammatical relation) information, such as sentence subject and object information. This information can be difficult to obtain when using other approaches such as constituent tree analysis and lexical rules [RP92, BvdR96]. The sentence role information offered by the Link Parser is also consistent with the observation by Gomez *et al.* [GSD99] that "the output of a parser should not be a tree, but a structure in which constituents are related to each other not by immediate dominance relations, but by simple dominance

relations", which is supported by the Description theory [MHF83, Mar87]. The reason is that syntax does not provide reliable evidence for the construction of immediate dominance relations between two constituents.

3.3.2 Typed dependency from the Stanford parser

Recently, de Marneffe *et al.* [dMMM06] proposed Typed Dependency (grammatical relation) based on the constituent tree output structure of the Stanford Parser [KM02] in order to promote the practical usage of the Stanford parser. The Typed Dependencies extracted from the constituent tree structured parsing results are convenient for practical natural language applications such as NLP-based conceptual data modeling.

Figure 20 shows the Typed Dependency hierarchy. The hierarchy contains 48 grammatical relations. The most general relation is *dependent* and can be further divided into *aux* (auxiliary), *arg* (argument), *mod* (modifier), etc. The parsing results are tagged with these tags. For instance, the Typed Dependency parse result for sentence "the company has 50 plants located in 40 states" is following,

```
det(company-2, the-1)
nsubj(has-3, company-2)
num(plants-5, 50-4)
dobj(has-3, plants-5)
partmod(plants-5, located-6)
num(states-9, 40-8)
prep_in(located-6, states-9)
```

Similar to the Link Grammar, the grammatical relations between different words such as nominal subject (*nsubj*), direct object (*dobj*), participial modifier (*partmod*), etc. are spelled out explicitly. Typed Dependencies also provide some semantic-oriented tags, such as *agent*. These grammatical relations can be utilized to extract entity relations for automated conceptual data modeling similar to the approach in [DM06].

dep - dependent
- aux - auxiliary
  - auxpass - passive auxiliary
  - cop - copula
- conj - conjunct
- cc - coordination
- arg - argument
  - subj - subject
    - nsubj - nominal subject
      - nsubjpass - passive nominal subject
    - csubj - clausal subject
  - comp - complement
    - obj - object
      - dobj - direct object
      - iobj - indirect object
      - pobj - object of preposition
    - attr - attributive
    - ccomp - clausal complement with internal subject
    - xcomp - clausal complement with external subject
    - compl - complementizer
    - mark - marker (word introducing an advcl)
    - rel - relative (word introducing a rcmod)
    - acomp - adjectival complement
  - agent - agent
- ref - referent
- expl - expletive (expletive *there*)
- mod - modifier
  - advcl - adverbial clause modifier
  - purpcl - purpose clause modifier
  - tmod - temporal modifier
  - rcmod - relative clause modifier
  - amod - adjectival modifier
  - infmod - infinitival modifier
  - partmod - participial modifier
  - num - numeric modifier
  - number - element of compound number
  - appos - appositional modifier
  - nn - noun compound modifier
  - abbrev - abbreviation modifier
  - advmod - adverbial modifier
    - neg - negation modifier
  - poss - possession modifier
  - possessive - possessive modifier (*'s*)
  - prt - phrasal verb particle
  - det - determiner
  - prep - prepositional modifier
- sdep - semantic dependent
  - xsubj - controlling subject

Figure 20: The Typed Dependency hierarchy [dMMM06]

3.4 HEURISTIC RULES

Heuristic rules have been utilized in previous research with the aim of capturing the relationships between different elements in natural languages and those in ERDs. The rules proposed in the following sections are based on the idea that "Nouns are corresponding to entities and attributes while verbs are likely relation candidates in ERDs". They are different from the rules in previous research in that these rules are based on higher level grammatical relations of two state of the art parsers (such resources were usually not available for much of the previous research) instead of word classes and related direct patterns.

Because in database requirements specifications sentences can be treated as independent units that specify either an entity to entity relation or an attribute attachment to an entity, here the entity relationships are extracted per sentence.

3.4.1 Heuristic rules based on link types

In this section, examples of heuristic rules based on link types generated from the Link Parser are proposed. These rules can extract most of the entity relationships that are specified in the requirements documents. The total number of rules required to cover all the sentential patterns in requirements documents may be very large, however, a small set of rules may cover a significant number of sentential patterns. Some more specific heuristic rules are discussed in Section 3.5. The heuristic rules are in the following format,

$$\textit{Link type conditions} \Rightarrow \langle \textit{relation} \quad \textit{entity}1 \quad \textit{entity}2 \rangle$$

The symbol"⇒"separates the left hand side (the conditions) and the right hand side (the suggested candidate entity relation tuple). The entity relation tuples extracted will be further processed in later stages.

A brief summarization of the semantic descriptions of the major link types used in the heuristic rules proposed in this section is in Table 2. A detailed documentation of these link types can be found at `http://www.link.cs.cmu.edu/link/dict/index.html`.

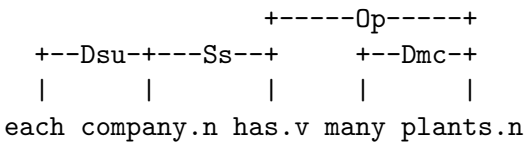
These link types provide rich information about the syntactic and partial semantic information of the underlying sentences. The following are some examples of the major

heuristic rules to cover different sentence structures. The full set of heuristics rules based on link types is in Appendix C.

3.4.1.1 Active sentence rules Rule (3.1) is one of the most general rules that cover the basic SVO sentence structure with a direct object. The subject ($S.LW$, the left word of the S link type) and Object ($O.RW$, the right word of the O link type) are extracted as the entity candidates and the verb ($S.RW$, the right word of the S link type) is extracted as the relation candidate. The $[sp]$ (single/plural) attribute is not significant in terms of identifying entities but can be used for evidence of cardinality. The "*" means there can be zero or more minor link types (lowercase letters) following the major link types (capitalized letters). Link type sequence is required on the left hand side of the rules.

$$S[sp]^* + O[sp]^* \;\Rightarrow\; \langle S.RW \quad S.LW \quad O.RW \rangle \tag{3.1}$$

For instance, given the following link parsing result,



The tuple $\langle has\ company\ plants \rangle$ can be extracted with rule (3.1).

This rule is based the general ideas from [Che83]. It is a combination and specification of Chen's rule 1 and rule 2:

> Rule 1: A common noun (such as "person", "chair") in English corresponds to an entity type in an ER diagram.
> Rule 2: A transitive verb in English corresponds a relationship type in an ER diagram.

Rule (3.1) is designed to extract the subject and object of a simple active sentence. The subject or the object may not be a common noun and hence not corresponding to an entity type. Disambiguation rules will further process the elements extracted from rule 3.1. The main verb of the sentence is proposed as a relationship type which is consistent with Chen's rule 2.

Table 2: A brief summarization of some of the link types

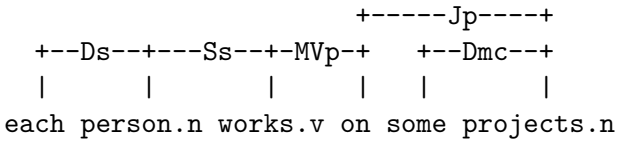
Link types	**Descriptions**
S	connects subject nouns to finite verbs
SF	is a special connector used to connect expletive subjects like "it" and "there" to finite verbs
RS	is used in subject-type relative clauses to connect the relative pronoun to the verb
O	connects transitive verbs to their objects, direct or indirect
MV	connects verbs and adjectives to modifying phrases that follow
M	connects nouns to various kinds of post-noun modifiers
J	connects prepositions to their objects
P	connects forms of the verb "be" to various words that can be its complements
I	connects infinitive verb forms to certain words such as modal verbs and "to"
PP	connects forms of "have" with past participles
TO	connects verbs and adjectives which take infinitival complements to the word "to"
OF	connects certain verbs and adjectives to the word "of"
CO	connects "openers" to subjects of clauses
R	connects nouns to relative clauses
MX	connects nouns to post-nominal noun modifiers
A	connects pre-noun adjectives to following nouns
AN	connects noun-modifiers to following nouns
D	connects determiners to nouns
N	connects the word "not" to preceding auxiliaries and modals

Many of the following heuristic rules are also based on these basic correspondence patterns between elements in natural language and elements of entity relationship diagrams.

Rule (3.2) covers the basic SVO sentence structure with a phrasal verb (prepositional verb). In this case, the subject ($S.LW$) and the objects of the preposition ($J.RW$) are extracted as the entity candidates. The phrasal verb together with the preposition ($S.RW$-$MV.RW$) is extracted as the relation candidate.

$$S[sp]^* + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle S.RW\text{-}MV.RW \quad S.LW \quad J.RW \rangle \tag{3.2}$$

For instance, given the following link parsing result,



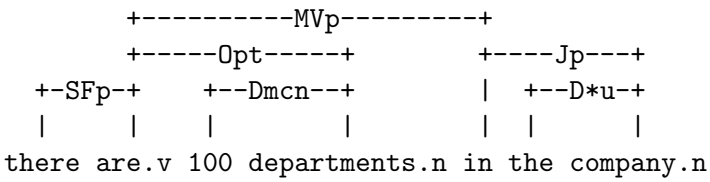
The tuple $\langle works\text{-}on\ person\ projects \rangle$ can be extracted with rule (3.2).

3.4.1.2 Expletive sentence rules In contrast to the previous rules where the subjects of the sentences are usually entity candidates, for expletive sentences, the objects of the preposition are more likely to be the entity candidates. The relation candidates vary in different cases.

In rule (3.3), the copular verb together with the preposition is proposed as the relation. Alternatively, the copular verb can be replaced by "*have/has*" (Chen's Rule 5 [Che83]).

$$SF[sp]^* + O[spt]^* + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle SF.RW\text{-}MV.RW \quad O.RW \quad J.RW \rangle \tag{3.3}$$

For instance, given the following link parsing result,



The tuple $\langle are\text{-}in\ departments\ company \rangle$ can be extracted with rules (3.3).

3.4.1.3 Passive sentence rules In database conceptual data modeling domain, relations are non-directional as shown in Figure 2. Although active voice statements can be used to represent almost any requirements, passive voice statements are also used frequently. The parsing structures of passive voice statements are different from the corresponding active sentence structures. Hence, different heuristic rules are proposed. Passive sentence structures have not been addressed explicitly in previous research perhaps because none of the previous heuristic rules were directly based on parsing results of different sentence structures. Of the multiple verbs in passive sentences, usually only one of them (which may not be the main verb) is extracted as the relation candidate based on the analysis of example sentences in the small database requirements documents corpus used.

Rule (3.4) covers basic passive sentence structures. It is the passive format of rule (3.2) with more conditional link types on the left hand side of the rule.

$$S[spx]^* + Pv + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle Pv.RW\text{-}MV.RW \quad S.LW \quad J.RW \rangle \tag{3.4}$$

For instance, given the following link parsing result,

```
    +--Spx-+---Pv--+--MVp--+---Jp---+
    |      |       |       |        |
plants.n are.v divided.v into departments.n
```

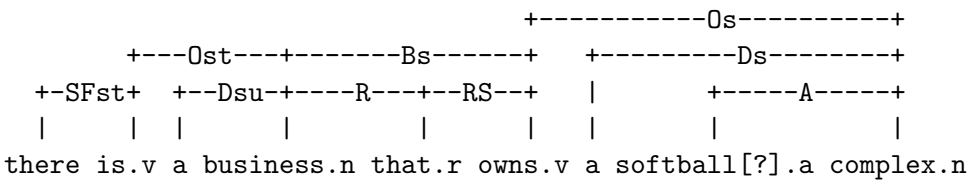
The tuple $\langle$*divided-into plants departments*$\rangle$ can be extracted with rule (3.4).

3.4.1.4 Subordinate clauses and other rules Entity relationships are specified not only by the main sentences, but also by the subordinate clauses and past participle modifiers. This is based on Chen's rule 10 "A clause in English is a high-level entity type abstracted from a group of interconnected low-level entity and relationship types in ER diagrams" [Che83]. However, the heuristic rules proposed here are incapable of distinguishing the hierarchical structure (high-level entity types vs. low-level entity types), which is more challenging to deal with, as stated in Chen's rule 10.

Rule (3.5) is an example rule that handles relative clauses. In this case, the reference can be resolved by replacing the subject of the relative clause with the object of the main clause as the entity candidate.

$$R + RS + O[sp]^* \;\Rightarrow\; \langle RS.RW \quad R.LW \quad O.RW \rangle \tag{3.5}$$

For instance, given the following link parsing result,

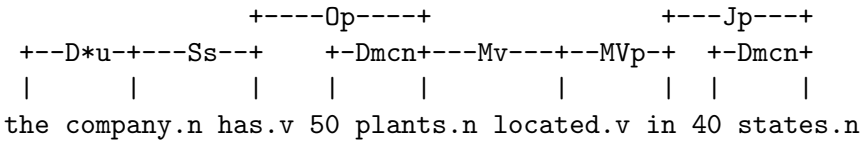


The tuple $\langle owns\ business\ complex \rangle$ can be extracted with rule (3.5).

Rule (3.6) extracts entity relation information from past participle modifiers.

$$Mv + MV[sp]^* + J[sp]^* \ \Rightarrow\ \langle Mv.RW\text{-}MV.RW \quad Mv.LW \quad J.RW \rangle \tag{3.6}$$

For instance, given the following link parsing result,



The tuple $\langle located\text{-}in\ plants\ states \rangle$ can be extracted with rule (3.6).

3.4.2 Heuristic rules based on Typed Dependencies

The heuristic rules proposed in the previous sections are based on the link types from the parsing results of the Link Parser. Theoretically, the entity relationships can be extracted from any parsing results or even without parsing as reported in some of the previous research. The major differences are relative difficulty, performance and scalability. In this section, some of the major heuristic rules based on the Typed Dependencies, which are derived from the constituent tree parsing results from the Stanford Parser, are discussed.

The major Typed Dependency tags used in the following heuristic rules for entity relationship extraction are {*nsubj, dobj, prep, nsubjpass, xcomp, agent, expl, partmod, rcmod, purpcl, amod, nn, neg*}. The semantics of these tags can be found in Figure 20.

The heuristic rules discussed in this section are similar to the ones based on link types and are intended to cover the same sentence structures. The same set of example sentences is used to illustrate the application of these rules compared with the corresponding rules in the previous sections. It is of interest to see that the heuristic rules proposed in this work

are based on high level grammatical relations regardless which specific parser is used, as long as the desired high level grammatical relations are present.

3.4.2.1 Active sentence rules Rule (3.7) covers the basic SVO sentence structure with a direct object as covered by rule (3.1). The left hand side of the rule (*nsubj*, the nominal subject; *dobj*, the direct object) are the conditional Typed Dependencies required to fire the rule. $w1, w2$ and $w3$ are numbered English words from the Typed Dependency parsing results. Word order sequence is required. The right hand side of the rule follows the same tuple structure as used in previous section.

$$nsubj(w2, w1) + dobj(w2, w3) \Rightarrow \langle w2 \quad w1 \quad w3 \rangle \tag{3.7}$$

For instance, given the following Typed Dependency parsing result,

```
each company has many plants.
```

```
det(company-2, each-1)                nsubj(has-3, company-2)
amod(plants-5, many-4)                dobj(has-3, plants-5)
```

The tuple $\langle has\ company\ plants \rangle$ can be extracted with rule (3.7).

Rule (3.8) covers a similar sentence structure as rule (3.7) but with a phrasal verb as the predicate. The required conditional Typed Dependencies are *nsubj* (nominal subject) , *prep* (prepositional modifier) and *pobj* (object of preposition).

$$nsubj(w2, w1) + prep(w2, w3) + pobj(w3, w4) \Rightarrow \langle w2\text{-}w3 \quad w1 \quad w4 \rangle \tag{3.8}$$

For instance, given the following Typed Dependency parsing result,

```
each person works on some projects.
```

```
det(person-2, each-1)                 nsubj(works-3, person-2)
prep(works-3, on-4)                   det(projects-6, some-5)
pobj(on-4, projects-6)
```

The tuple $\langle works\text{-}on\ person\ projects \rangle$ can be extracted with rule (3.8).

3.4.2.2 Expletive sentence rules Rule (3.9) covers the basic expletive sentence structure as rule (3.3). The required conditional Typed Dependencies are *expl* (expletive "there"), *nsubj* (nominal subject), *prep* (prepositional modifier) and *pobj* (object of the preposition).

$$expl(w2, w1) + nsubj(w2, w3) + prep(w3, w4) + pobj(w4, w5) \;\Rightarrow\; \langle w2\text{-}w4 \quad w3 \quad w5 \rangle \quad (3.9)$$

For instance, given the following Typed Dependency parsing result,

```
there are 100 departments in the company.
```

```
expl(are-2, there-1)                    num(departments-4, 100-3)
nsubj(are-2, departments-4)             prep(departments-4, in-5)
det(company-7, the-6)                   pobj(in-5, company-7)
```

The tuple $\langle$*are-in departments company*$\rangle$ can be extracted with rule (3.9).

3.4.2.3 Passive sentence rules Rule (3.10) covers one of the basic passive sentence structures. The required conditional Typed Dependencies are *nsubjpass* (passive nominal subject), *prep* (prepositional modifier) and *pobj* (object of preposition).

$$nsubjpass(w2, w1) + prep(w2, w3) + pobj(w3, w4) \;\Rightarrow\; \langle w2\text{-}w3 \quad w1 \quad w4 \rangle \quad (3.10)$$

For instance, given the following Typed Dependency parsing result,

```
plants are divided into departments.
```

```
nsubjpass(divided-3, plants-1)          auxpass(divided-3, are-2)
prep(divided-3, into-4)                 pobj(into-4, departments-5)
```

The tuple $\langle$*divided-into plants departments*$\rangle$ can be extracted with rule (3.10).

3.4.2.4 Subordinate clauses and other rules Rule (3.11) is a rule that deals with relative clauses as covered by rule (3.5). The required conditional Typed Dependencies are *rcmod* (relative clause modifier), *nsubj* (nominal subject) and *dobj* (direct object).

$$rcmod(w1, w2) + nsubj(w2, w3) + dobj(w2, w4) \Rightarrow \langle w2 \quad w1 \quad w4 \rangle \tag{3.11}$$

For instance, given the following Typed Dependency parsing result,

```
there is a business that owns a softball complex.
```

```
expl(is-2, there-1)                    det(business-4, a-3)
nsubj(is-2, business-4)                nsubj(owns-6, that-5)
rcmod(business-4, owns-6)              det(complex-9, a-7)
amod(complex-9, softball-8)            dobj(owns-6, complex-9)
```

The tuple $\langle owns\ business\ complex \rangle$ can be extracted with rule (3.11).

Rule (3.12) is a rule that extracts entity relation information from past participle modifiers as covered by rule (3.6). The required conditional Typed Dependencies are *partmod* (participle modifiers), *prep* (prepositional modifier) and *pobj* (object of preposition).

$$partmod(w1, w2) + prep(w2, w3) + pobj + (w3, w4) \Rightarrow \langle w2\text{-}w3 \quad w1 \quad w4 \rangle \tag{3.12}$$

For instance, given the following Typed Dependency parsing result,

```
the company has 50 plants located in 40 states.
```

```
det(company-2, the-1)                  nsubj(has-3, company-2)
num(plants-5, 50-4)                    dobj(has-3, plants-5)
partmod(plants-5, located-6)           prep(located-6, in-7)
num(states-9, 40-8)                    pobj(in-7, states-9)
```

The tuple $\langle located\text{-}in\ plants\ states \rangle$ can be extracted with rule (3.12).

The full set of heuristic rules based on Typed Dependency is in Appendix D.

3.4.3 Other dependency parsers

The heuristic rules proposed in previous sections are based on the dependency-related Link Parser and the Typed Dependency from the Stanford Parser. The choice of these two parser was made based on the correspondence between higher grammatical relations in the parsing results and elements in the target formalisms[DM06]. Dependency parsing results are preferred over constituency (phrase-structure) parsing results because dependency tags are close to the semantic relationships used in conceptual data modeling. However, this is not true for all the dependency parsers. Dependency parsing, which describe how words link to their arguments, has a long history [Hud84]. The resurgence of interest in dependency parsing lately is mainly on multi-lingual parsing and statistic-based training. Actually, dependency parsers are frequently used for free word order languages such as Russian and German [MP06]. Several other dependency parsers have been examined for their applicabilities in conceptual data modeling.

MSTParser. MSTParser is a recently developed multilingual dependency parser. The parsing algorithm is based on searching for maximum spanning trees (MSTs) in directed graphs. It has been evaluated on many corpora and for English the reported accuracy is 91.5% [MP06]. The strong points of this parser lie in parsing multiple languages with one model, especially those free word languages such as German.

MiniPar. MiniPar is a broad coverage parser for English. An evaluation on the SUSANNE corpus shows that MiniPar achieves about 88% precision and 80% recall with respect to dependency relationships [Lin98].

Compared with the Link Parser and the Typed Dependency from the Stanford Parser, MSTParser [MP06], MiniPar [Lin98] as well as other dependency parsers such as MaltParser [NN05] are not suitable for automated conceptual data modeling because the desired higher level grammatical relations are not presented in the parsing results. As in the case of constituency (phrase structure) parsing, significant post-parsing processing is needed to extract entity relationship tuples from these dependency parsing results. In addition, these recent dependency parsers are still in their early development stages and are not as mature as the Link Parser and the Stanford Parser. The Link Parser was first implemented in 1991

and the latest version is 2001. The Stanford Parser was first implemented in 2002 and the latest version is 2007.

3.4.4 Discussion

The heuristic rules proposed in this work are based on some of the basic transformation ideas from Chen's work[Che83]. Although several earlier research projects have also extended Chen's work to some extent, the heuristic rules proposed here are significantly different from them. These rules are constructed directly from the high level grammatical relations from dependency and related parsing results with focus on different sentence structures. They are more specific and operational. They can be easily translated into a machine executable format.

The heuristic rules proposed here are more specific and operational than Chen's eleven general heuristic rules [Che83]. Chen provided some general guidelines on how to translate English sentences to ERDs. Chen's rule 1, 2 and 5 were covered explicitly in this work, but based on more specific parsing results than general word class information. Chen's rules (6 – 11) are partially covered in some of the rules proposed in this work. However, Chen's rule 3 and 4 were not used. It is true that "an adjective in English corresponds to an attribute of an entity in an ER diagram" (Chen's rule 3) as in "red car". However, adjectives are not commonly used as attributes in database requirements. The most often used attributes such as "name", "phone", "address", "ssn", etc. are nouns. The heuristic rules proposed in this works are focused on attribute nouns and entity nouns.

The heuristic rules proposed here are less lexicalized than Omar's rules [OHM04][2]. Many specific words such as "number, code, date, birth, name", etc. were used in Omar's rules. The heuristic rules proposed in this work are based on links or typed dependencies which are more general and have higher coverage than specific words in terms of input scope. However, the utilization of domain knowledge in this work is related to Omar's lexicalized heuristic rules.

The heuristic rules proposed here are also quite different from the rules in [RP92] and

[2]Only eight example rules could be found in the paper.

[GSD99]. Rolland & Prox [RP92] adopted a kind of integrated syntactic and semantic approach. The thematic role information used in their grammatical rules is related to the high level grammatical relations proposed in this work. However, the syntactic information was tightly coupled with the semantic representations in their rules. It is well known that such integrated approaches require greater development efforts and the resulting grammars tend to be brittle because of the complexity involved. Similarly, Gomez's rules [GSD99] also integrated sematic and syntactic information in the underlying natural language understanding module.

3.5 SOME SPECIAL ISSUES IN HEURISTIC RULES

Heuristic rules based on link types and Typed Dependencies such as those proposed in previous section (also see Appendix C and Appendix D) are the major extraction rules. However, there are some special sentence structures and local issues that need to be addressed by refining or modifying the results extracted by these rules.

3.5.1 Of structures

An "Of structure" is defined as a noun phrase modified by prepositional phrases containing the preposition "of" as illustrated in the following two sentences[3],

(a). *Each employee of the department works on at least one project.*
(b). *A number of employees belong to each department.*

The parsing results of these two sentences from the Link Parser are:

```
(a)
          +-------------Ss-------------+
          |      +-----Js----+         |     +--------Js--------+
  +---Ds--+--Mp--+  +---Ds---+         +-MVp-+  +IDB+-EN-+--Ds--+
  |       |      |  |        |         |     |  |   |    |      |
each employee.n of the department.n works.v on at least one project.n
```

[3]The "of structure" discussed here does not include participles such as in "consist of".

```
(b)
      +-----------Sp----------+     +-----Js-----+
+--Ds-+--Mp-+--Jp--+          +-MVp-+   +---Ds---+
|     |     |      |          |     |   |        |
a number.n of employees.n belong.v to each department.n
```

For sentence (a), rule (3.2) applies and yields $\langle works\text{-}on\ employee\ projects \rangle$. The result is correct. The Link Parser treats the noun before "of" as the head of the "of structure". This is true in general. However, for sentence (b), the same heuristic rule (3.2) applies and yields $\langle belong\text{-}to\ number\ department \rangle$, instead of $\langle belong\text{-}to\ employees\ \ department \rangle$. The rule itself is not wrong, but the parser is incapable of distinguishing different "of structures" and labels the word "number" as the subject (Sp link here). It is hard to determine which noun in the "of structure" should be treated as an entity based solely on syntactic information. It requires semantic analysis to solve this problem. The Link Parser is a syntactic parser and is incapable of handling such a semantic issue. At the heuristic rule level, the following three refining rules are used to propose an alternative tuple for each "of structure". The alternative entities proposed with these rules will be disambiguated at a later stage.

$$S[sp]^* + M[sp]^* + J[sp]^* \wedge M.RW = of \wedge \langle relation \quad S.LW \quad entity2 \rangle \Rightarrow \langle relation \quad J.RW \quad entity2 \rangle \tag{3.13}$$

For the above sentence (a), rule (3.13) applies and extracts $\langle works\text{-}on\ department\ \ projects \rangle$. For example sentence (b), the same rule applies and extracts $\langle beglong\text{-}to\ employee\ department \rangle$.

Rule (3.14) extracts an alternative tuple for the object of a preposition.

$$J1[sp]^* + M[sp]^* + J2[sp]^* \wedge M.RW = of \wedge \langle relation \quad entity1 \quad J1.RW \rangle \Rightarrow \langle relation \quad entity1 \quad J2.RW \rangle \tag{3.14}$$

For instance, given the following link parsing result,

```
 +-Dmc-+--Sp--+--Ix-+---Pv--+--MVp--+-J+-Mp+  +--Dmc-+
 |     |      |     |       |       |  |   |  |      |
the tasks.n can.v be.v performed.v at any of the plants.n
```

An alternative tuple $\langle performed\text{-}at\ tasks\ plants \rangle$ can be extracted.

Rule (3.15) extracts an alternative tuple for the direct object of a sentence.

$$O[sp]^* + M[sp]^* + J[sp]^* \wedge M.RW = of \wedge \langle relation \quad entity1 \quad O.LW \rangle$$
$$\Rightarrow \langle relation \quad entity1 \quad J2.RW \rangle \qquad (3.15)$$

For instance, given the following link parsing result,

```
                                  +-----Os-----+
    +--Spx--+---Pv--+--TO--+--I--+      +--Ds-+-Mp-+--Jp--+
    |       |       |      |     |      |     |    |      |
doctors.n are.v allowed.v to perform.v any kind.n of diagnosis.n
```

An alternative entity tuple $\langle perform\ doctors\ diagnosis \rangle$ can be extracted with rule (3.15).

The same problem exists for Typed Dependency parsing. The Typed Dependency parsing results of the above two example sentences are:

```
(a) Each employee of the department works on at least one project.

    det(employee-2, Each-1)                 nsubj(works-6, employee-2)
    prep(employee-2, of-3)                  det(department-5, the-4)
    pobj(of-3, department-5)                prep(works-6, on-7)
    quantmod(one-10, at-8)                  dep(at-8, least-9)
    num(project-11, one-10)                 pobj(on-7, project-11)

(b) A number of employees belong to each department.

    det(number-2, A-1)                      nsubj(belong-5, number-2)
    prep(number-2, of-3)                    pobj(of-3, employees-4)
    prep(belong-5, to-6)                    det(department-8, each-7)
    pobj(to-6, department-8)
```

For sentence (a), rule (3.8) applies and extracts $\langle works\text{-}on\ employee\ project \rangle$. For sentence (b), the same rule applies and extracts $\langle belong\text{-}to\ number\ department \rangle$.

In order to deal with "of structures", rules similar to (3.13, 3.14, and 3.15) are needed. Rule (3.16) extracts alternative entities for the tuples extracted by rule (3.8).

$$nsubj(w4, w1) + prep(w1, of) + pobj(of, w3) \wedge \langle w4\text{-}p \quad w1 \quad w5 \rangle$$
$$\Rightarrow \langle w4\text{-}p \quad w3 \quad w5 \rangle \qquad (3.16)$$

For the above example sentence (a), rule (3.16) applies and extracts $\langle works\text{-}on\ department\ project\rangle$. For example sentence (b), the same rule applies and extracts $\langle belong\text{-}to\ employee\ department\rangle$.

Similarly, rules (3.17, 3.18, 3.19) suggest alternative entity relation tuples for other "of structures".

$$nsubjpass(w4, w1) + prep(w1, of) + pobj(of, w3) \wedge \langle w4\text{-}p \quad w1 \quad w5\rangle \Rightarrow \langle w4\text{-}p \quad w3 \quad w5\rangle \tag{3.17}$$

$$dobj(w2, w3) + prep(w3, of) + pobj(of, w4) \wedge \langle w2\text{-}p \quad w1 \quad w3\rangle \Rightarrow \langle w2\text{-}p \quad w1 \quad w4\rangle \tag{3.18}$$

$$prep(w2, w3) + pobj(w3, w4) + prep(w4, of) + pobj(of, w5) \wedge \langle w2\text{-}w3 \quad w1 \quad w4\rangle \Rightarrow \langle w2\text{-}w3 \quad w1 \quad w5\rangle \tag{3.19}$$

3.5.2 Coordinating conjunctions

Coordinating conjunctions are used frequently in natural language. In database requirements, coordinating conjunction indicates multiple entity relationships. For instance, the sentence "the company has 7 plants and 3,000 employees" should be translated into two entity relation tuples $\langle has\ company\ plants\rangle$ and $\langle has\ company\ employees\rangle$. Coordinating conjuction can happen at the subject, predicate or object position.

The Link Parser has special mechanisms for automatically transforming the majority of coordinating conjunctions to separate sub-linkages on which the previous link types based rules work. For instance, the Link Parser gives two sub-linkages for the sentence "the company has 7 plants and 3,000 employees",

```
                +----Op---+
 +--D*u-+---Ss--+   +-Dmcn+
 |      |       |   |     |
the company.n has.v 7 plants.n and 3,000 employees.n
```

```
                +--------------Op-------------+
 +--D*u-+---Ss--+                    +--Dmcn--+
 |      |       |                    |        |
the company.n has.v 7 plants.n and 3,000 employees.n
```

Rule (3.1) can deal with both sub-linkages and extracts $\langle has\ company\ plants \rangle$ and $\langle has\ company\ employees \rangle$. However, for Typed Dependency, the parsing result is,

```
the company has 7 plants and 3,000 employees.

det(company-2, the-1)                    nsubj(has-3, company-2)
num(plants-5, 7-4)                       dobj(has-3, plants-5)
cc(plants-5, and-6)                      num(employees-8, 3,000-7)
conj(plants-5, employees-8)
```

Rule (3.7) applies but can only extract $\langle has\ company\ plants \rangle$. To handle the conjunctions, the following five rules are needed,

$$cc(w2, w1) + conj(w2, w3) \wedge \langle w4\text{-}p \quad w2 \quad w5 \rangle \Rightarrow \langle w4\text{-}p \quad w3 \quad w5 \rangle \tag{3.20}$$

$$cc(w3, w4) + conj(w3, w5) \wedge \langle w2\text{-}p \quad w1 \quad w3 \rangle \Rightarrow \langle w2\text{-}p \quad w1 \quad w5 \rangle \tag{3.21}$$

$$nsubj(w2, w1) + cc(w2, w4) + conj(w2, w5) + prep(w5, w6) + pobj(w6, w7) \wedge \langle w2\text{-}w6 \quad w1 \quad w3 \rangle \Rightarrow \langle w5\text{-}w6 \quad w1 \quad w7 \rangle \tag{3.22}$$

$$nsubj(w2, w1) + cc(w2, w4) + conj(w2,w5) + dobj(w5, w6) \wedge \langle w2 \quad w1 \quad w3 \rangle \Rightarrow \langle w5 \quad w1 \quad w6 \rangle \tag{3.23}$$

$$preconj(w4, w3) + cc(w4, w5) + dep(w4,w6) \wedge \langle w2\text{-}p \quad w1 \quad w4 \rangle \Rightarrow \langle w2\text{-}p \quad w1 \quad w6 \rangle \tag{3.24}$$

3.5.3 Compound noun phrases

Compound noun phrases are usually indicative of entity or attribute types in database requirements. In the previous rules, only the head nouns are proposed as entity candidates. For instance, for the sentences "Full-time students need to register for at least three courses. Part-time students may register for only one course.", the previous rules only extract "student" as an entity candidate, but not the pre-noun modifiers "Full-time" and "Part-time" which are semantically significant in this case. In other cases, the pre-noun modifiers may not be semantically significant but make the tuples extracted more natural and understandable. For instance, for the sentence "Each work task is associated with a job type", a tuple $\langle associated\text{-}with\ task\ type \rangle$ will be extracted. However word "type" is not a good indicator for an entity type. It is better to include the noun-modifiers with the nouns together as entity candidates, which will produce $\langle associated\text{-}with\ work\text{-}task\ job\text{-}type \rangle$.

Both the Link Grammar and Typed Dependencies provide syntactic tags for pre-noun modifiers. The "A" link type which connects pre-noun adjectives to following nouns and "AN" link type which connects noun modifiers to following nouns are indicators of compound noun phrase. Correspondingly, there are "*amod*" and "*nn*" tags in Typed Dependencies.

Noun modifiers should be extracted together with the entity nouns. Rules (3.25, 3.26) can be used to add noun modifiers to the entity nouns extracted by the previous rules.

$$nn(w2, w1) \wedge \langle w3\text{-}p \quad w2 \quad w4 \rangle \Rightarrow \langle w3\text{-}p \quad w1\text{-}w2 \quad w4 \rangle \tag{3.25}$$

$$nn(w4, w3) \wedge \langle w2\text{-}p \quad w1 \quad w4 \rangle \Rightarrow \langle w2\text{-}p \quad w1 \quad w3\text{-}w4 \rangle \tag{3.26}$$

Adjective pre-noun modifiers can also be extracted in two similar rules. However, it is questionable whether adjective pre-noun modifiers should be used in the entity names. For instance, in the case of "several students", "several" is a quantifier and should not be used in entity names while in the case of "full-time students", "full-time" is a restriction modifier and is appropriate to be used in entity names especially in the context of distinguishing from "part-time students".

3.5.4 Relationship cardinality rules

The cardinality of a relationship in ERD is the number of instances of one entity participating with the other entity in the relationship. It is usually described in the following three basic forms, 1:1, 1:N and M:N.

Each of the relationships in an ERD should have a cardinality for each of the connected entities. Ideally, the cardinality of a relationship should be reasoned from the semantics of the database requirements. However, the single/plural format of the noun entities and the verb as well as the adjective and cardinal pre-noun modifiers provides syntactic indications for relationship cardinality extraction. Especially useful are the pre-noun modifier of the object of a sentence.

One type of pre-noun modifier is the cardinal number such as "500" in the sentence "the plants have 500 employees". Because the number of cardinal numbers is infinite, the POS tag or other tags such as "num" in Typed Dependencies can be used to identify the cardinal number pre-noun modifiers. Another type of pre-noun modifiers is quantifier, such as "many", "some", "several", etc. A small set of quantifiers can capture most of the cases used in database requirements.

If there are no cardinal numbers or quantifiers as pre-noun modifiers or the pre-noun modifiers don't provide the desired cardinality information, the single/plural format of the entity nouns can be used to infer the cardinality information as in the sentence "students take courses".

But there are some exceptional cases, such as in "each student takes at least one course". The single format of the word "course" and even the numerical modifiers "one" suggest a "1" cardinality, but in this case, it is "many".

The most difficult problem in extracting relation cardinality is the implicit or common sense statements in database requirements. It is not unusual to see one-way cardinality statements in database requirements documents. For instance, in the sentence "each student must take at least one course", it can be inferred that the cardinality between the relation "take" and entity "course" should be "many". Human readers may have no difficulty to see that there are usually more than one student for a course, however, this is not stated in the

requirements. A ideal database requirements document should state the relation in both ways, such as "each student takes at least one course and each course can be taken by many students". A simple solution to this problem is to use default cardinality and prompt user's correction at a later stage.

3.5.5 Ternary relations

Each of the previous extracted entity relation tuples is a binary relation. Ternary relations can be obtained by combining two relevant binary relations with the following rule,

$$\langle w2\text{-}p1 \quad w1 \quad w3\rangle \wedge \langle w2\text{-}p2 \quad w1 \quad w4\rangle \Rightarrow \langle w2\text{-}p \quad w1 \quad w3 \quad w4\rangle \tag{3.27}$$

For instance,

```
employees perform work tasks at work stations.
```

```
nsubj(perform-2, employees-1)          nn(tasks-4, work-3)
dobj(perform-2, tasks-4)               prep(perform-2, at-5)
nn(stations-7, work-6)                 pobj(at-5, stations-7)
```

Rule 3.7 and rule 3.8 as well the compound noun rules (3.25, 3.26) apply and extract tuples $\langle perform\ employees\ work\text{-}tasks\rangle$ and $\langle perform\text{-}at\ employees\ work\text{-}stations\rangle$. Then rule 3.27 applies and generates a ternary relation tuple $\langle perform\text{-}at\ employees\ work\text{-}tasks\ work\text{-}stations\rangle$ as shown in Figure 21.

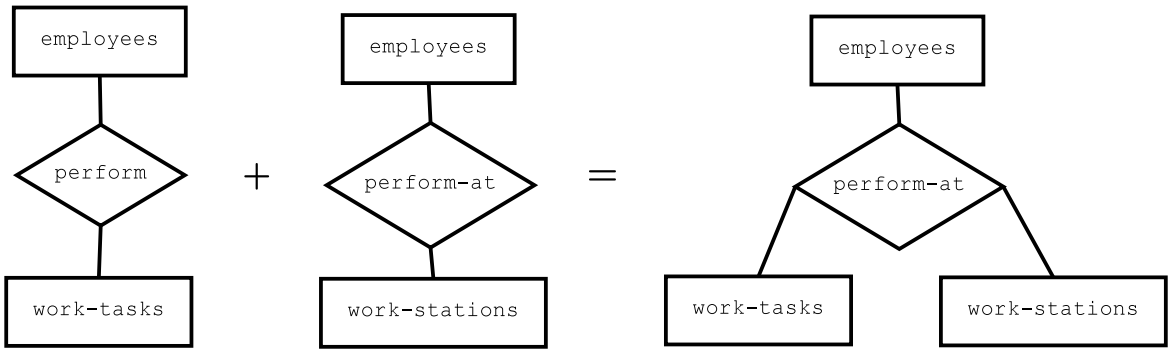


Figure 21: A ternary relation

3.5.6 Deduplication[4]

In database requirements documents, the same entity relationship may be stated several times with different word form variations. Some of the variations may only involve single/plural or tense while others may involve synonyms, abbreviation or paraphrase. As a consequence, multiple tuples with little variation may be extracted from a requirements document. It is desirable to deduplicate the entity relationship tuples extracted by the heuristic rules, especially when merging the different tuples extracted from different sentences into the final ERD.

- **Base form based deduplication.** Using the base form of the entity, relation and attribute words can deduplicate many variations that only involve single/plural and tense, e.g. "employees" vs "employee", "sell" vs. "sold".
- **Synonym or antonym based deduplication.** From time to time, database requirements writers may use different synonyms or antonyms to reiterate the same requirements. For instance, "students buy books" vs. "students purchase books", or "students buy books" vs. "books are sold to students". Usually, human readers don't have difficulty to understanding such reiteration. A synonym dictionary such as WordNet can be used to deduplicate the entity relationship tuples in this case.
- **Abbreviation or paraphrase based deduplication.** Deduplication also needs to be performed on abbreviation and paraphrase in some cases. For instance, in the sentence, "There are five worker unions represented in the company, and every employee belongs to exactly one union", the last word "union" represents the same entity as "worker unions". It is much more difficult to deduplicate the entity relationship tuples systematically at this level. Users intervention may be more appropriate.

3.5.7 Resolution principle

A problem in applying the previous heuristic rules is that multiple rules may apply to the same sentence and yield conflicting entity relationship tuples. For instance, rule (3.7) applies

[4]Similar to the usage of this term in data storage processing, here the term refers to the process of removing duplicate entity relationship tuples extracted by the heuristic rules.

to any sentence to which rule (3.11) applies. Looking at the rules, it is not hard to find that the left hand side (preconditions) of rule (3.7) is a sub-part of rule (3.11). To solve this problem, the following resolution principle is used,

Specificity principle. *If two heuristic rules are applicable to the same parsing structure, the more specific one has higher priority.*

A rule is more specific than another one if it has more preconditions on the left hand side of the rule than the other one. According to this definition, rule (3.11) is more specific than rule (3.7), so rule (3.11) preempts rule (3.7).

3.6 ENTITY RELATIONSHIP DISAMBIGUATION

Over-generation and incorrect attachment are the most serious problems that need to be addressed at the semantic analysis phase. For automated conceptual data modeling, a mechanism that disambiguates entity nouns from attribute nouns and evaluates the likelihood of a relation between two entities is needed. For instance, if the previous heuristic rules proposed one tuple $\langle$*located-in plants employees*$\rangle$, for a human being it is trivial to judge that it is unlikely that relation *"located-in"* holds between entities *"plants"* and *"employees"*. However, it is notoriously difficult to enable a computer program to answer such questions. It is a common sense reasoning problem and has been investigated intensively in artifical intelligence related research fields, but no satisfactory general solutions are available yet. So the following proposed approaches explore the question of how much accuracy can be obtained using state of the art semantic resources and relatively simple procedures in semi-structured translation problems.

3.6.1 WordNet as a lexical filter

WordNet has been explored as a tool to disambiguate nouns in automated database conceptual data modeling [DM06]. Although entities are described with nouns, the correspondence between entities and nouns is not perfect since nouns are also used to refer

to many concepts that are not usually considered entities but rather are abstractions that help describe the nature of the situation but do not refer to the sorts of "things" that are represented as entities in ERDs. Nouns such as "number", "set" or "type" are examples of such abstractions. In fact, it is not easy to distinguish which nouns can identify entities and which can not. Such distinctions can depend on the context and a human's ability to apply general world knowledge. In order to implement automated disambiguation, the hypernym chains in WordNet have been utilized to disambiguate entities (in the ERD sense) from non-entities. Two classes of hypernym chains have been identified. One is for nouns such as "set, list, range, type, kind, unit, $\cdots$", the hypernym chain of which usually leads to {*magnitude, property, attribute, abstraction*}. If evidence of this class of hypernym chains is found, the noun will not be treated as an entity. The other class of hypernym chains is for nouns such as "student, teacher, course, company, plant, $\cdots$", the hypernym chain of which usually leads to {*organization, object, physical entity, group, event, act*}. If evidence of this class of hypernym chains is found, then the noun will be treated as an entity. This method can also be used to disambiguate entities from attributes.

Table 3 provides a top taxonomy of English nouns according to the top ontology in WordNet. The often used nouns list includes 2,998 noun lemmas taken from Word Frequency in Written and Spoken English – based on the British National Corpus [LRW01]. The link-noun list includes 14,354 nouns taken from the Link Parser Lexicon. The WordNet noun list includes 116,956 nouns taken from the WordNet dictionary. Roughly speaking, about 60% of the nouns in English can be categorized as entity nouns and 30% of the nouns can be categorized as attribute nouns according to the WordNet top ontology.

These top noun categories can be used as a general standard to disambiguate attribute nouns from entity nouns, and it can also be utilized to handle the "of-structure" problem discussed in Section 3.5.1. Only one of the alternative entity relationship tuples extracted from an "of-structure" should be presented in the final entity relationship representation. One way to utilize the above noun categories is illustrated in the following rule which states that if there are two entity relationship tuples extracted from one "of-structure" and one of the conflict nouns belongs to the attribute noun category while the other belongs to the entity noun category, then the tuple with the noun belonging to attribute noun category should be

removed. This rule is useful to handle "of structures" such as "a number of student".

$$\langle r\ e1\ e2\rangle \wedge \langle r\ n1\ e2\rangle \wedge e1 \in \textit{attribute-nouns} \wedge n1 \in \textit{entity-nouns} \Rightarrow \textit{remove}\langle r\ e1\ e2\rangle \quad (3.28)$$

Table 3: English noun category WordNet vs. ER

CDM Category	**WN Category**	**Often Used Nouns 2,998**	**LP Nouns 14,354**	**WN Nouns 116,956**
entity nouns	object substance group physical process thing event	63.5%	67.8%	63.5%
attribute nouns	cognition attribute measure communication	31.6%	27.8%	21.0%
others	others	4.9%	4.4%	13.7%

3.6.2 The Web corpus and search facilities as a semantic filter

The Web corpus and search facilities was explored as a semantic filter for disambiguating coordinating conjunctions and prepositional phrase attachment. The approach was similar to the utilization of word cooccurrence in various NLP applications, such as speech recognition and word sense disambiguation [DPL94]. The prepositional phrase attachment problem is reflected as an entity attachment problem in entity relationship extraction. For instance,

for the sentence *"The company has 50 plants located in 40 states and approximately 100,000 employees"*, the following 3 tuples can be extracted with the heuristic rules proposed in Section 3.4.1 based on the first parse of the Link parsing results.

1.1. <has.v, company.n, plants.n>

1.2. <located.v-in, plants.n, states.n>

1.3. <located.v-in, plants.n, employees.n>

Because this sentence is quite complicated, the first parse from the Link parser (or the Stanford parser) is preferred syntactically but is semantically wrong. In fact, the third one is correct (see Appendix E). In order to improve the coverage, multiple parses were utilized which might lead to overgeneration. The refining rules, the WordNet lexical filter, and a semantic filter were used to address such problems. For instance, the following tuples can be extracted from the second and the third parse of the Link Parser results,

2.1. <has.v, company.n, plants.n>

3.1. <has.v, company.n, plants.n>

3.2. <located.v-in, plants.n, states.n>

3.3. <has.v, company.n, employees.n>

Tuple 1.3 and Tuple 3.3 are conflict tuple pairs based on the assumption that for a given sentence only one parse is correct. The semantic implication of Tuple 1.3 is "plants located in employees". Obviously, it does not make sense. One way to solve this problem is to post a question to ask the user. However, an automated disambiguation method is certainly preferred. In an earlier paper [DM06], a shallow quasi-statistical approach based on empirical evidence found in the Web text corpus to justify the likeliness of relationship among terms was described. This was done by generating SOAP calls to a web search engine (Google, Yahoo! and Live Search all offer similar services) to determine how frequently the phrases utilizing the terms in the tuples (in that literal form with variant forms, as illustrated in Table 4) are found in the web corpus. It is clear that the web corpus contains a large number of expressions indicative of the meaning of Tuple 3.3, and no case reflecting the content of Tuple 1.3, thus providing evidence that the third parse result which produced Tuples 3.1, 3.2 and 3.3 is the correct interpretation of this sentence.

Table 4: Queries to disambiguate a relationship attachment

Tuple 1.3 queries	hits	Tuple 3.3 queries	hits
“plant located in employee”	0	“company has employee”	405
“plant located in employees”	0	“company has employees”	954
“plants located in employee”	0	“companies have employee”	485
“plants located in employees”	0	“companies have employees”	706
sum	0	sum	2550

The above approach was extended in two dimensions. One approach was to expand the items utilized in the exact phrase searching from the simple plurality expansion in Table 4 to a wider range of additional simple sentences. A small sentence generation module was built to generate simple sentences such as the following from the given tuples.

```
a plant is located in an employee
a plant is located in employees
plants are located in an employee
plants are located in employees
a plant was located in an employee
a plant was located in employees
plants were located in an employee
plants were located in employees
a plant may be located in an employee
a plant may be located in employees
plants may be located in an employee
plants may be located in employees
a plant could be located in an employee
a plant could be located in employees
plants could be located in an employee
plants are located in an employee
......
```

Obviously, there are many such simple-structured sentences. The sum of the number of hits returned by a search engine over all these simple sentences was used to disambiguate the conflicting tuples.

The other extension tried was to do a sentence-based search[5]. It was based on the assumption that the number of sentences that contain a given tuple[6] provides good evidence to justify the likelihood of the relationship held between the two entities in the tuple. Currently, there are no search engines or other resources that provide sentence-based search on a large enough corpus. One preliminary solution is to issue key word queries based on a given tuple to a search engine and extract the contents of a number of links that are returned by the search engine and then process the extracted contents to see whether there are sentences that contains the given tuple. However, this approach requires much longer time to disambiguate a given tuple because it involves extracting text contents from the Internet, although it can be expedited to some extent by extracting the cached contents from the search engine's database instead of going to the real web pages directly.

3.6.3 Other possible approaches

3.6.3.1 A ConceptNet approach An alternative to the previous web corpus-based approach is to build a semantic network that stores all the likelihood values between different concepts and relations. The immediate difficulty of such a project is the vastness and complexity of natural language concepts and the possible relations held between them. Nevertheless, there are already some valuable resources that can be utilized. ConceptNet is a commonsense knowledge base that was mined from the MIT Open Mind Commonsense corpus. The knowledge base is a semantic network consisting of over 1.6 million assertions of common sense knowledge encompassing the spatial, physical, social, temporal, and psychological aspects of everyday life [LS04].

The ConceptNet knowledge base is structured in a relatively simple, easy-to-use semantic network. Although ConceptNet doesn't provide the desired APIs for automated conceptual data modeling applications, the knowledge base of the system can be easily accessed. A specific module was built upon the ConceptNet knowledge base to answer questions like "what other concepts are related to a target concept?", "what is the most likely relation held

[5]that is a search to find sentences that contain all the words of a triple.

[6]Ideally, the relation and two entities in a tuple correspond to the predicate, the subject and the object of a sentence.

between two concepts?". For instance, if the concept "*student*" is given, the module provides a set of concepts that are conceptually related to "*student*" ordered by some likelihood as shown following,

```
ConceptuallyRelatedTo    : student    person               109
ConceptuallyRelatedTo    : student    teacher              93
ConceptuallyRelatedTo    : student    randy                89
ConceptuallyRelatedTo    : student    test                 74
ConceptuallyRelatedTo    : student    israel               60
ConceptuallyRelatedTo    : student    class                43
ConceptuallyRelatedTo    : student    school               40
ConceptuallyRelatedTo    : student    that change be josh  31
ConceptuallyRelatedTo    : student    group                28
CapableOf                : student    learn                26
ConceptuallyRelatedTo    : student    that change be randy 24
ConceptuallyRelatedTo    : student    homework             22
ConceptuallyRelatedTo    : student    library              20
CapableOfReceivingAction : student    teach                19
ConceptuallyRelatedTo    : student    stick                15
ConceptuallyRelatedTo    : student    grade                13
CapableOf                : student    study                13
ConceptuallyRelatedTo    : student    shane                12
ConceptuallyRelatedTo    : student    difference           12
......
```

If two entities, for example *student* and *book*, are given, the ConceptNet can provide the possible relations held between these two concepts, such as *CapableOf: student, read book*. However, it has the sparse data problem in terms of practical usage in entity relationship disambiguation.

3.6.3.2 Cyc as a knowledge source Cyc is a well-known large scale knowledge source and a formalized representation of a vast quantity of fundamental human knowledge: facts, rules of thumb and heuristics for reasoning about objects and events of everyday life [LG90]. It is potentially usable as a knowledge source for entity relationship disambiguation for automated conceptual data modeling in several ways.

Cyc can be used to disambiguate compound noun phrases. Cyc contains more than 163,000 constants and most of them are compound noun phrases. The problem of whether a pre-noun modifier (either an adjective or noun) of a compound noun phrase should be included as a part of an entity type name in the final formalisms as discussed in Section

3.5.3 can be resolved by checking whether the compound noun phrase is a Cyc constant. If a compound noun phrase is a Cyc constant, there is a good reason to treat the whole compound noun phrase as an entity instead of just the head of the noun phrase because the Cyc constants are created based on some solid considerations by human cyclists. For instance, #$AccountType, #$WorkStation, #$CreditCard and #$ShoppingCart are all Cyc constants. On the other hand side, compound noun phrase #$CertainSkill is not a Cyc constant, which suggests that "Skill" instead of "CertainSkill" should be used as an entity name.

Cyc can also be used for entity relationship reasoning in a way similar to the use of ConceptNet discussed in the previous section. Cyc contains relations like #$conceptuallyRelated and can answer similar questions as the ConceptNet. For instance, Cyc can answer the query:

EL Query: (#$*conceptuallyRelated* #$*Student* ?*What*) in HumanSocialLifeMt

by giving #$CourseOfStudy.

However, it is a rather complicated process to use the Cyc knowledge base for entity relationship reasoning. First, the constants in Cyc represent word senses instead of the lexical words. For instance, the word "course" corresponds to three Cyc constants: #$CourseOfAMeal, #$CourseOfStudy and #$Path-Generic. To facilitate the mapping between English words and Cyc constants, Cyc introduced about 18,000 lexical constants such as #$Course-TheWord and #$Student-TheWord. Each of these lexical constants has a "denotation" slot which provides the corresponding Cyc constants that relate to the lexical word. For instance, the following query shows the denotation slots of the lexical word "course".

```
Mt : (MtSpace GeneralEnglishMt AnytimePSC)
EL Query : (denotation Course-TheWord ?POS ?NUM ?SENSE)
Status : Suspended, Exhaust Total
Answer          ?POS     ?NUM ?SENSE
*[Explain #2] CountNoun 0 CourseOfStudy
*[Explain #1] CountNoun 1 CourseOfAMeal
*[Explain #0] CountNoun 2 Path-Generic
```

Most of the general Cyc constants have a "genStringAssertion" slot that maps Cyc constants back to english words. Second, relations specified in Cyc are more specific than in

ConceptNet. There are more than 12,000 relations defined in Cyc compared with only 20 relations defined in ConceptNet [LS04]. So it is quite challenging to select the right relations to explore in Cyc.

Another application of Cyc for entity relationship reasoning is to explore its selectional restrictions for entity attachment resolution. Taking the tuple $\langle$*located-in plants employees*$\rangle$ as an example, the fundamental question that needs to be answered is whether it is possible that plants are located in employees. The selectional restriction here is that plants (either the botany sense or the industrial plant sense) are usually not located in employees except in some unusual contexts. Cyc has the mechanism to specify the argument types of relations. For instance, it requires that the first argument of relation #$buys must be a #$SocialBeing (specified by #$arg1Isa #$buys #$SocialBeing) and the second argument must be #$SomethingExisting (specified by #$arg2Isa #$buys #$SomethingExisting). So the entity relationship tuple $\langle$*buys students books*$\rangle$ is valid according to Cyc. On the contrary, tuple $\langle$*buys software books*$\rangle$ is not valid according to Cyc.

Similar to the application of ConceptNet to entity relationship reasoning, Cyc also has the sparse data problem. Much of the general common sense knowledge is not represented in Cyc yet.

3.6.3.3 Integrated syntactic and semantic approach The above approaches utilize external large knowledge resources for semantic disambiguation. It is also possible to use an integrated syntactic and semantic approach as described in [All95]. In previous research in conceptual data modeling, Rolland & Prox [RP92] and Gomze *et. al.* [GSD99] adopted some kind of integrated syntactic and semantic approaches (see section 2.1.3). The most demanding tasks for such an approach involve defining the semantics (logic forms) of each individual word and lots of semantic augmented syntactic rules to parse the inputs. Although the requirements documents exhibit some kinds of the regularities, the development and computation of such meaning structures are impossible unless in a very restricted domain.

3.6.4 CDM Domain knowledge-based disambiguation

The approaches discussed above are characterized by exploring large scale knowledge resources to disambiguate entities, attributes and relations. Such approaches usually have good coverage, however, the resources used are not specific to database requirements applications. So they may not be efficient, especially in some specific application domains. Domain specific knowledge in conceptual data modeling was used to address some specific problems. For instance, a small set of typical entity nouns such as {*student, company, department, manager, employee,* $\cdots$}, a typical non-entity set such as {*name, size, year, idea, time, each, some, number,* $\cdots$} and a set of typical attribute nouns such as {*name, id, salary, age, address,* $\cdots$} [7] were used first to disambiguate some of the entity or attribute nouns before consulting the top noun categories in WordNet. The set of words was built from a small database requirements specification corpus. If the application domain is known, a more specific set of typical entity nouns and attributes nouns can be built.

3.7 THE TARGET FORMALISM

For automated conceptual data modeling, a module capable of translating entity relationship tuples extracted from those heuristic rules to ERDs is needed. An open source automatic graph drawing package Graphviz [GN00] was explored for ERD generation. The Graphviz package provides many useful features for different diagrams drawing, such as options for colors, fonts, tabular node layouts, line styles, hyperlinks and customizing shapes. The Graphviz layout programs require that the input file be written in DOT language, a plain text graph description language.[8] For instance, the following desired entity relationship tuples,

```
<has company plant>
<located-in plant state>
<has company employee>
<divided-into plant department>
```

[7] overlapping with the set of typical attribute nouns

[8] The DOT language BNF definition is at http://www.graphviz.org/doc/info/lang.html.

```
    <subdivided-into department work_station>
    <are-in department company>
    <are-in work-station company>
    <associated-with work-task job-type>
    <performed-at job-type plant>
    <perform employee work_task>
    <performed-at work-task plant>
    <represented-in union company>
    <belong-to employee union>
```

would need to be translated into a DOT language description such as the following,

```
graph ER{
  node [shape=box]; company; plant; state; employee; department; work_station;
                    work_task; job_type; union;
  node [shape=diamond, style=filled, color=lightgrey]; has; located_in;
       divided_into; subdivided_into; are_in; associated_with; performed_at;
       perform; perform_at2; belong_to;
  company -- has; has -- plant;
  plant -- located_in; located_in -- state;
  company -- has; has -- employee;
  plant -- divided_into; divided_into -- department;
  department -- subdivided_into; subdivided_into -- work_station;
  department -- are_in; are_in -- company;
  work_station -- are_in; are_in -- company;
  work_task -- associated_with; associated_with -- job_type;
  job_type -- performed_at; performed_at -- plant;
  employee -- perform; perform -- work_task;
  work_task -- perform_at2; perform_at2 -- work_station;
  union -- represented_in; represented_in -- company;
  employee -- belong_to; belong_to -- union;
}
```

The above DOT language description is essentially a text representation of a standard graph $G = (V, E)$. The entities are specified in box nodes and the relations are specified in diamond nodes. For each of the tuples, two edges are specified. For instance, the tuple $\langle has\ company\ plant \rangle$ corresponds to "company – has; has – plant;" in DOT. This example illustrates a simplified, relatively straightforward translation from tuples to DOT descriptions. In the system implemented, feature-rich DOT descriptions including attributes, cardinality, colors to show the system confidence levels, etc. are generated by a DOT-language Generator module. With the DOT descriptions, ERD diagrams such as Figure 22 can be automatically generated with the *dot* layout engine. The final diagram can be

exported to various formats such as jpg, png, eps, tiff, etc. An interactive user interface was implemented in a java applet based on the Grappa package.

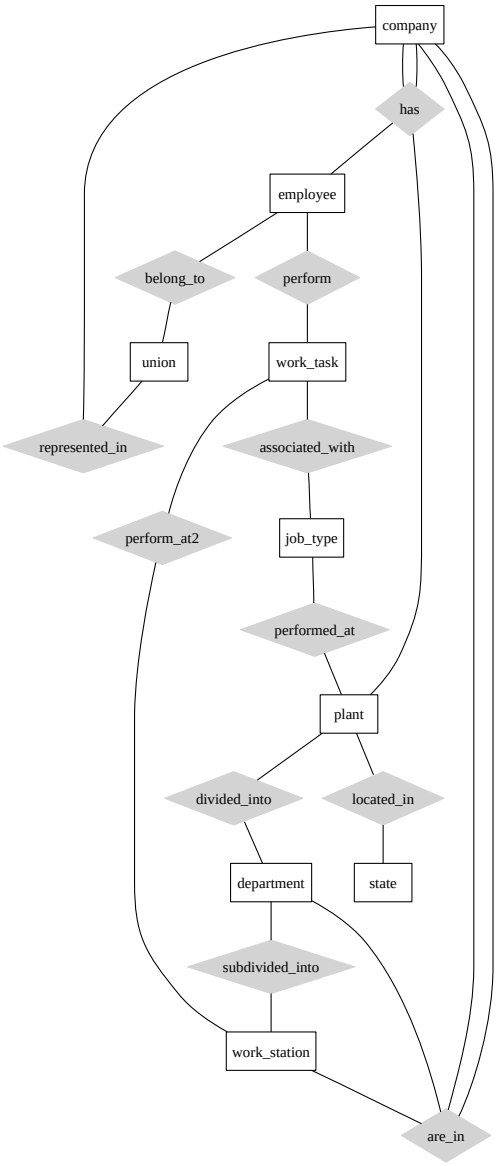


Figure 22: An ERD example generated by Dot

4.0 PROTOTYPE ARCHITECTURE

A pipe line style of web-based automated conceptual data modeling system was implemented based on several prototype components built previously.

4.1 SYSTEM MODULES OVERVIEW

The system modules and data flow are shown in Figure 23. There are three types of modules: core modules, system specific modules and support modules. All of the core modules and the system specific modules were designed and implemented specifically for this ACDM system, while the support modules were adapted from the corresponding open-source resources.

The system passes natural language database requirements specifications (DRS) as inputs to a preprocessing module. The major functionality of the preprocessing is to canonicalize these inputs based on domain knowledge. After preprocessing, a Sentencifier module is used to sentencify the inputs based on the assumption that sentences are relatively independent statement units in database requirements. Then the sentences are split into two parts according to the sub-language characteristics of the inputs using a pattern matching approach. The Attribute Extractor module takes the Stanford Tagger results, domain knowledge and words base forms from WordNet as inputs, and produces a list of attributes for some entities or relations. All other sentences that are not handled by the Attribute Extractor module are passed to the Stanford Parser. The Facts Generator produces the proper facts for the ER Extractor module based on the parsing results and the word base forms from WordNet. The heuristic rules are implemented in CLIPS in the ER Extractor module and generate a set of entity-relation or entity-attribute tuples. Also domain-based

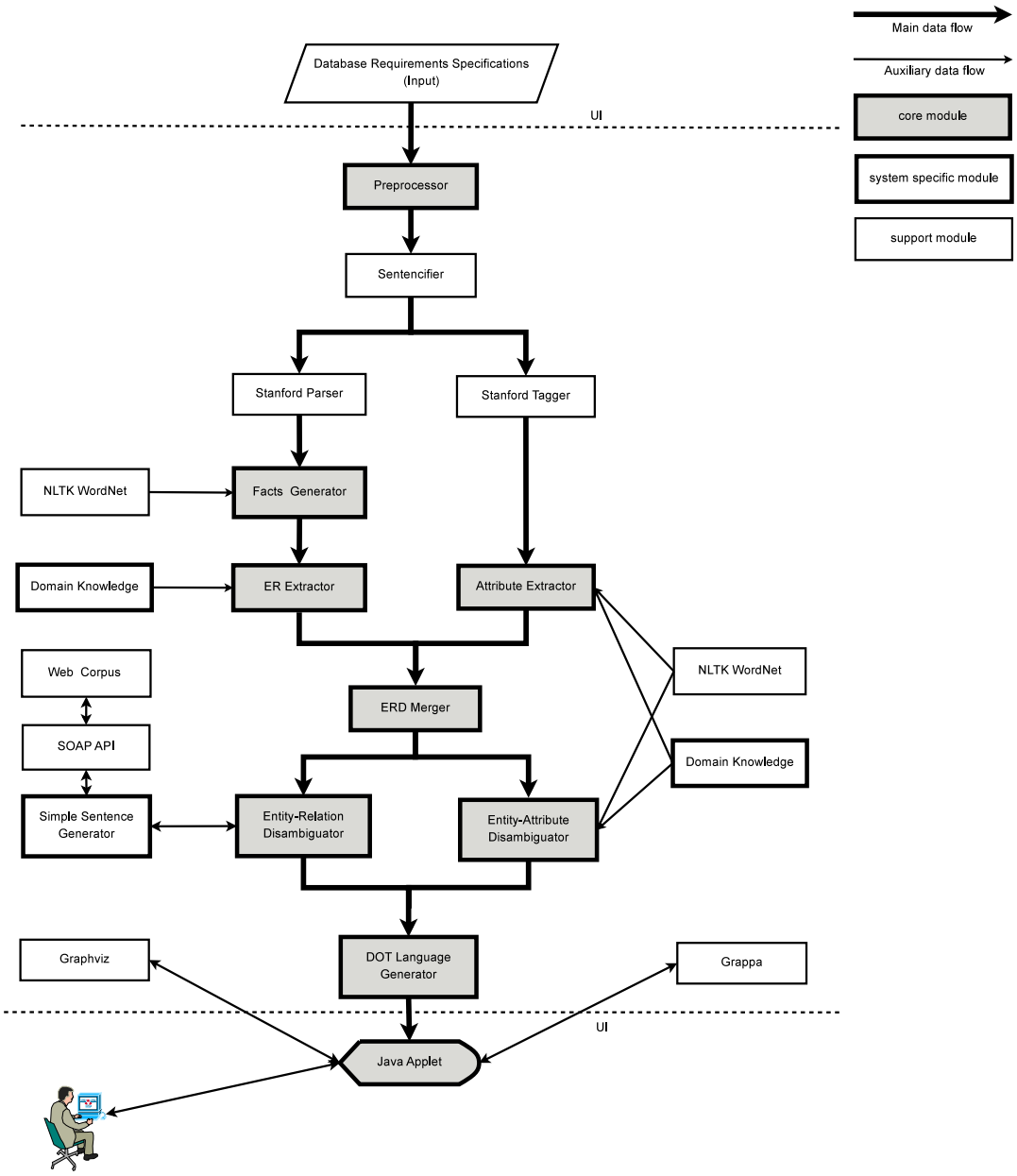


Figure 23: System modules and data flow

entity relationship disambiguation is performed in the ER Extractor module. ERD Merger module merges the outputs from the Attribute Extractor module and the ER Extractor module. The Entity-Attribute Disambiguator module further disambiguates all the entities and attributes based on the top noun categories and hypernym chains from WordNet and assigns a confidence score for each of them. The Entity-Relation Disambiguator further disambiguates some of the relations produced by the ER Extractor based on the empirical evidence found in the Web corpus via the Simple Sentence Generator, which produces exact string queries based on a given entity relation tuple, and the SOAP APIs of a web search engine. The DOT Language Generator produces the text representation of the final ERDs in DOT language. The Java Applet draws the actual diagrams based on the DOT language inputs, the layout info from the Graphviz package and the Graphical elements from the Grappa package. Users can modify and manipulate the diagrams produced by the system in various ways with the interactive interface.

Compared with other system structures in the literature [Lee03, GSD99], this system architecture involves more third-party tools and large-scale knowledge resources and hence the components have been loosely coupled. However, the production system style rule extraction module is more robust and powerful in reasoning. This is the first time that some of the general resources such as typed dependency, DOT language and Graphviz have been used for automated conceptual data modeling. The CLIPS implementation of the heuristic rules is also unique to this system architecture.

4.1.1 Input user interface

In order to make the use of the system as simple as possible, a web-based input user interface as shown in Figure 24 is provided. Actually, the input interface consists of only one text input area and a submit button. Although no specific restrictions are required on the input texts, the desirable inputs are well-written grammatically correct database requirements specifications.

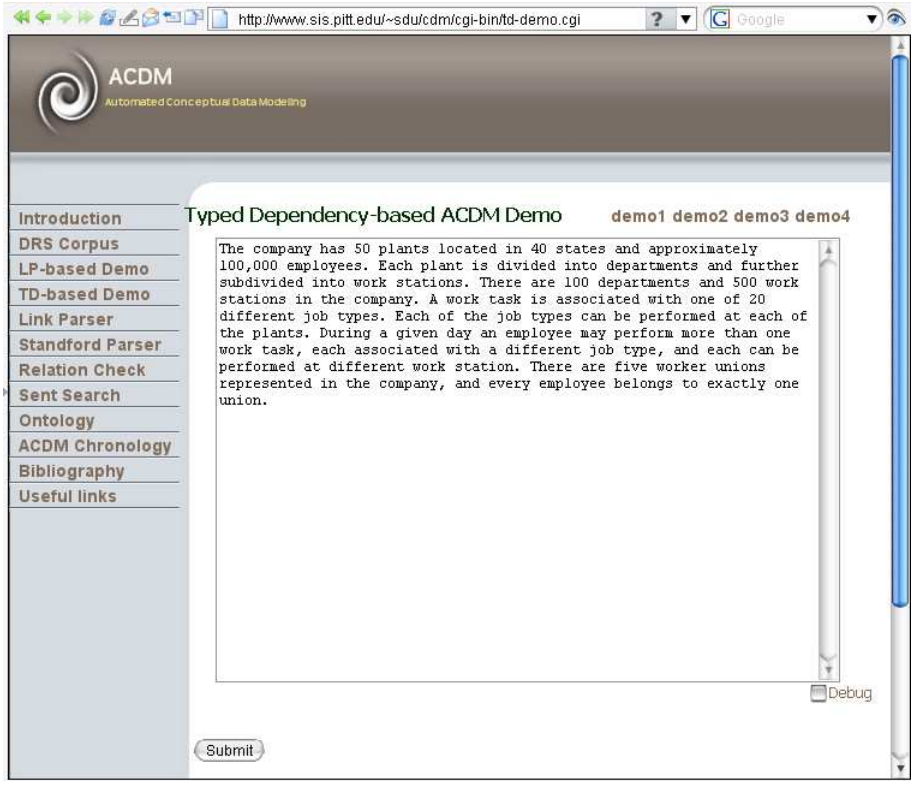


Figure 24: The web input interface of the ACDM system

4.1.2 Preprocessing

Preprocessing is usually very specific and application- and input-dependent. The functionalities of the preprocessing module of this implementation are:

- **Partial explanation statements removal.** Explanation statements in requirements aim at clarifying some of the concepts to the readers. They are helpful for human readers to understand the requirements better but useless or even harmful for automated requirements analysis. For instance, in "Each account has a unique number, a balance, and an account type (e.g. checking, savings, etc.)", the explanation part inside the parenthesis is not necessary to transform the statement into an ERD representation as shown in Figure 25. Sometimes the explanation statements pose challenges to NLP parsers as well as post parsing analysis. Heuristic rules based on parenthesis and

characteristic words ("*e.g., such as*") are utilized to remove the explanation statements during the preprocessing.

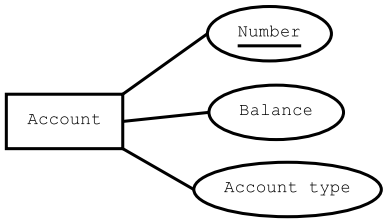


Figure 25: An example of NL-ER translation with preprocessing

- **Concatenation and canonicalization** To facilitate both parsing and post-parsing analysis, some of the domain phrases are concatenated into single words and some of the abbreviations are canonicalized in the preprocessing module. Table 5 illustrates some of the domain phrases. Taking the sentence "Each instructor has his/her SSN, first name and last name" as an example,

  (a) `Each/DT instructor/NN has/VBZ his\/her/NN SSN/NN ,/, first/JJ name/NN and/CC last/JJ name/NN ./.`

  (b) `Each/DT instructor/NN has/VBZ his/PRP$ or/CC her/PRP$ SSN/NN ,/, first/JJ name/NN and/CC last/JJ name/NN ./.`

  (a) without canonicalizing pronoun "his/her", the Stanford parser tagged "his/her" as a noun[1], while (b) with the canonicalization, the pronoun "his/her" was tagged correctly. In addition, the concatenation of some of the domain phrases supports the entity relationship cardinality analysis. The cardinality information can be obtained from both the single/plural format of the entity nouns and the determiners before nouns. However, there are exceptions. For instance, in the sentence "each student takes at least one course", the single format of the entity noun "course" as well as the bigram "one course" suggests a "1" cardinality for entity "course" which is wrong. N-gram[2] is feasible but

[1] with the NN tagger, the Attribute Extractor module will treat "his/her SSN" as an attribute instead of "SSN".

[2] 4-gram is needed here to handle the phrase "at least one course".

more complicated. However, with the concatenation, the single word *at-least-one* is used as a specific phrase to extract a "many" cardinality. The concatenation of domain phrase also improves the readability of the final conceptual data models.

4.1.3 Attribute extraction

The Attribute Extractor module was implemented in a pattern matching approach. A typical long enumerated attribute statement in database requirements documents is in this form:

```
entity (relation) X has attribute A, B, ..., E and F.
```

The Stanford NLP parser used does not perform well on the coordination conjunction and the internal structures of the attribute elements. A pattern matching approach based on regular expressions with POS tagged inputs was adopted to extract attributes from long enumerative sentence structures as well as some other similar attribute sentence structures. Table 6 shows the regular expressions used. All of the input sentences are matched against these regular expressions. Those sentences that are not covered by any of these regular expressions are piped to the Stanford Parser.

For each of the matched entity, relation or attribute elements (X, A, B, ...), a separate functional module determines the appropriate words to be extracted based on the POS information and the domain knowledge. Heuristic rules are used to extract the entity, relation and attribute names because it is tricky to determine how many words should be used to represent names without causing ambiguities in the context. For instance, in the requirement "Each concert has a concert ID ..... Each composition has a composition ID.", a human database designer would probably only use "ID" as an attribute name for entity "concert" instead of using "concert ID" which is redundant. However, in the sentence "Each check out transaction has a check out date, an expected return date and an actual return date.", each of the three attribute names consists of three English words. If only the last noun is used here, then three ambiguous "date" attributes will be generated for the relation "check out". Using long attribute names is not a good general solution, because it often results in ugly esthetics, e.g. "subjects" vs. "subjects they are currently teaching". The heuristic rules also

Table 5: Concatenation of domain phrases

more than one	more-than-one
at least one	at-least-one
at least two	at-least-two
one or more	one-or-more
one and only one	one-and-only-one
no more than	no-more-than
zero or more	zero-or-more
more than two	more-than-two
more than three	more-than-three
first name	first-name
last name	last-name
zip code	zip-code
part time	part-time
full time	full-time
phone number	phone-number
email address	email-address
social security number	social-security-number
id number	id-number
year of birth	year-of-birth
mailing address	mailing-address
area code	area-code
order number	order-number
item number	item-number
id number	id-number
check in	check-in
check out	check-out
his/her	his or her

Table 6: Regular expressions for attribute extraction

(.*)(have|has|identified)/VB[DGNPZ]?(.*),/,\s*which/WDT\s*\w*/\w* \s*?(.*)and/CC(.*)
(.*)(has|have|include|includes|contain|contains|consist|consists) /VB[DGNPZ]?(.*),/,\s*(\w+.*)and/CC(.*)
(.*)(has|have|include|includes|contain|contains|consist|consists) /VB[DGNPZ]?(.*),/,\s*(\w+.*),/,(.*)
([F|f]or/IN.*),/,.*(store|stores|record|records|keep|keeps|need|needs) (.*),/,\s*(\w+.*)and/CC(.*)
([F|f]or/IN.*),/,(.*),/,\s*(\w+.*)and/CC(.*)(stored|recorded|kept)
data/NNS about/IN (.*) (are/VBP)(.*),/,\s*(\w+.*)and/CC(.*)
(store|stores|record|records|keep|keeps|need|needs)(.*/NN.*)\'s/POS (.*),/,\s*(\w+.*)and/CC(.*)
(.*)(stored|recorded|kept|decribed|represented)(.*),/,\s*(\w+.*) and/CC(.*)
(.*)\'s/POS (information.*following.*):/:(.*),/,\s*(\w+.*)and/CC(.*)

deal with compound noun phrases such as "phone number", "one or more address" ,"year of birth", "social security number", etc.

It is true that the regular expressions listed in Table 6 don't guarantee to extract all the attributes in database requirements documents. However, one of the goals of the attribute extraction module is to split some of the hard parsing, long enumerative attribute sentences from others and pipe all sentences that are not covered by these regular expressions to the Stanford parser for further processing.

4.1.4 Typed dependency parsing

As discussed in Section 3.4, the parsing results from different parsers can be used. A prototype system was implemented based on the Link Parser results. However, it was found that the Stanford Parser was more robust and the format of the typed dependency grammatical relations was compatible with the requirement of this application. So in this implementation, the Stanford Parser was used. Although the Stanford Parser can be trained on a customized corpus, no tagged database requirements specification corpus is currently available. So the parser uses the standard English lexicalized PCFG trained on the Wall Street Journal from the Penn Treebank. To reduce the time delay of loading the PCFG grammars, a customized module was designed to run the parser as a server on a specific port.

4.1.5 Entity relationship extraction

4.1.5.1 CLIPS CLIPS (C Language Integrated Production System) is a rule-based expert system building language [GR98]. Although the heuristics rules proposed in section 3.4 can be implemented in other computer programming languages, CLIPS provides the most convenient and efficient way to implement those heuristic rules.

A CLIPS program consists of a set of facts and rules. (CLIPS also includes a full object-programming capability but that was not utilized in this project.) Facts can be simple logic-like lists of a relation and arguments, or they can be more object-like structures with a set of slots for each object and potential constraints on the fillers for those slots. The latter "unordered" facts (the slots can appear in any order when defining instances) were used in this project. Rules are typically (but not exclusively) used to represent heuristic knowledge in production systems, and they are used for that purpose in this project.

4.1.5.2 CLIPS Facts Each of the typed dependency parsing results is represented as a fact in CLIPS. The template definition of a typed dependency is as following,

```
(deftemplate td
  "elements of typed dependencies template"
  (slot td_name
        (type SYMBOL))
  (slot word1
        (type STRING))
  (slot position1
        (type NUMBER))
  (slot word2
        (type STRING))
  (slot position2
        (type NUMBER))
  (slot sent_id
        (type NUMBER)))
```

The "td_name" slot is to store the typed dependency name; the "word1" slot is to store the first word in the typed dependency tuple; the "position1" slot is to store the word position in a sentence starting from 1, similarly for slot "word2" and "position2"; the "sent_id" slot is to store the sentence id.

The basic word information for each word of the input texts is also represented as a fact. The template definition is as following,

```
(deftemplate unit
  "word information template"
  (slot word
        (type STRING))
  (slot root
        (type STRING))
  (slot pos
        (type SYMBOL))
  (slot position
        (type NUMBER))
  (slot sent_id
        (type NUMBER)))
```

The "word" slot is to store the original word string appearing in the input; the "root" slot is to store the base form of the word according to the *morphy* function from WordNet[3]; the "pos" is to store the POS taggers from the Stanford tagger; the "position" slot is to store the word position in a sentence starting from 1; the "sent_id" is to store the sentence id from

[3] The morphy function in WordNet produces wrong base forms for a small set of words, e.g. the base form of word "stations" is still "stations" from WordNet.

the input which is used to guarantee that only typed dependencies from the same sentence can match against each others.

The extracted entity relationship tuples (both intermediate and final results) are represented in facts according to the following template,

```
(deftemplate tuple
  "An entity relationship tuple template"
  (slot relation
        (type STRING))
  (slot relation_position
        (type NUMBER))
  (slot relation_base
        (type STRING))
  (slot relation_supplement
        (type STRING))
  (slot relation_supplement_position
        (type NUMBER))
  (slot entity1
        (type STRING))
  (slot entity1_position
        (type NUMBER))
  (slot entity1_base
        (type STRING))
  (slot entity1_cardinality
        (type SYMBOL))
  (slot entity1_supplement
        (type STRING))
  (slot entity1_supplement_position
        (type NUMBER))
  (slot entity2
        (type STRING))
  (slot entity2_position
        (type NUMBER))
  (slot entity2_base
        (type STRING))
  (slot entity2_cardinality
        (type SYMBOL))
  (slot entity2_supplement
        (type STRING))
  (slot entity2_supplement_position
        (type NUMBER))
  (slot relation_dup
        (type STRING))
  (slot rule
        (type SYMBOL))
  (slot sid
        (type NUMBER)))
```

The "relation" slot is to store the original word string appearing in the input extracted as relation (the verb); the "relation_position" slot is to store the word position in a sentence starting from 1; the "relation_base" is to store the base form of the relation word which is used for base form based deduplication; the "relation_supplement" is to store the possible preposition following a phrasal verb that should be extracted together with the verb as the suggested relation name; the "entity1_supplement" slot is to store the possible pre-noun modifier that should be extracted together with the noun as the suggested entity name; the "entity1_cardinality" slot is to store the cardinality information.

4.1.5.3 CLIPS Rules Each of the entity relationship extraction heuristic rules proposed in Section 3.4 was implemented as a CLIPS rule. A rule in CLIPS consists of an antecedent or left-hand side (LHS) and a consequent or right-hand side (RHS). The LHS of a rule is a set of conditions expressed as patterns which must be satisfied based on the existence or non-existence of the specific facts in the fact list for the rule to be fired. The inference engine in CLIPS provides the mechanism to atomatically match patterns against the current state of the fact list and determine which rule to fire according to the conflict resolution strategy.

The following are three example CLIPS rules corresponding to heuristic rule (3.7, 3.9, 3.10).

```
(defrule nsubj_dobj
  "extract subject-direct object tuple, rule(3.7)"
  (td (td_name nsubj) (word1 ?v1) (position1 ?p1) (word2 ?n1)
      (position2 ?p0) (sent_id ?s1))
  (td (td_name dobj)  (word1 ?v1) (position1 ?p1) (word2 ?n2)
      (position2 ?p2) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (entity1 ?n1) (entity1_position ?p0)
                 (entity2 ?n2) (entity2_position ?p2)
                 (rule a0) (sid ?s1)))
)
```

```
(defrule expl_nsubj_prep_pobj
  "extract expletive tuple, rule (3.9)"
  (td (td_name expl)  (word1 ?v1) (position1 ?p1) (word2 ?w1)
                      (position2 ?p0) (sent_id ?s1))
  (td (td_name nsubj) (word1 ?v1) (position1 ?p1)
                      (word2 ?n1) (position2 ?p2) (sent_id ?s1))
  (td (td_name prep)  (word1 ?n1) (position1 ?p2)
                      (word2 ?w2) (position2 ?p3) (sent_id ?s1))
  (td (td_name pobj)  (word1 ?w2) (position1 ?p3)
                      (word2 ?n2) (position2 ?p4) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3 ?p4))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (relation_supplement ?w2) (relation_supplement_position ?p3)
                 (entity1 ?n1) (entity1_position ?p2)
                 (entity2 ?n2) (entity2_position ?p4)
                 (rule e0) (sid ?s1)))
)
```

```
(defrule nsubjpass_prep_pobj
  "extract passive subj prep pobj tuple, rule (3.10)"
  (td (td_name nsubjpass) (word1 ?v1) (position1 ?p1) (word2 ?n1)
                          (position2 ?p0) (sent_id ?s1))
  (td (td_name prep)      (word1 ?v1) (position1 ?p1) (word2 ?w1)
                          (position2 ?p2) (sent_id ?s1))
  (td (td_name pobj)      (word1 ?w1) (position1 ?p2) (word2 ?n2)
                          (position2 ?p3) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (relation_supplement ?w1) (relation_supplement_position ?p2)
                 (entity1 ?n1) (entity1_position ?p0)
                 (entity2 ?n2) (entity2_position ?p3)
                 (rule p0) (sid ?s1)))
)
```

Besides the basic heuristic extraction rules, all of the rules that address the specific issues discussed in Section 3.5 were also implemented in CLIPS rules, see Appendix F.

4.1.6 Entity attribute disambiguation

The entity attribute disambiguation was performed according to the top noun category and the hypernym chains from WordNet for each entity. Three top level category sets are defined as following,

- strong-entity = ("group", "physical_object", "physical_entity", "thing")
- mid-entity = ("substance","event", "communication", "physical_process")
- weak-entity = ("cognition", "attribute", "measure", "constituent", "language_unit")

If the hypernym chain of an entity reaches to one of the categories in the "strong-entity" set, the system will label the entity with a high confidence score. The cases for the "mid-entity" set and "weak-entity" set are similar. Different colors were used to indicate the different confidence scores in the final ERDs generated. Users are encouraged to make judgments regarding the entities with low confidence scores.

4.1.7 Entity relation disambiguation

The entity relation disambiguation was performed by searching for the number of appearances of the exact form of short sentences built from entity relationship tuples or triples from the Google web corpus via the Google search SOAP APIs[4]. A small natural language generation module was built to generate the exact search queries for each case. Exact phrase search may lead to many false negative cases because of the sparse data problem, so only the highly dubious relations such as the entity relationship tuples generated from the coordinating conjunction rules were disambiguated in this approach. A lower confidence score will be assigned to a relation if the various short sentences built from an entity relation tuple don't appear in the Google web corpus. Different colors were also used to indicate the different confidence scores in the final ERDs generated. Users are encouraged to take actions on the relations with low confidence scores.

[4] Yahoo! SOAP search APIs were also tried, however, the performance was not as good as that of Google.

4.1.8 DOT language generation

In order to use the Graphviz package to draw the final ERD diagram, the entity relationship information extracted was translated into the DOT language representation. A DOT file consists of two major parts: the node specifications and the edge specifications. Three types of nodes are used in ERD: "box" for entities, "ellipse" for attributes and "diamond" for relations. The translation from a tuple representation of ERDs to a DOT representation is relatively straightforward. However, there are some complicated situations. For instance, attribute names need to be annotated properly because different entities can have same attribute names in ERDs while duplicate nodes with same names are not allowed in DOT representations.

4.1.9 Final ERD representation

The Graphviz package provides some tools to generate various graphic representation of ERDs. However, there are some shortcomings of the Graphviz package,

- It is hard to modify the diagrams generated.
- Users have little control of the layouts generated by the Graphviz layout rendering engine.
- The Graphviz is a general graphic package, not specific to ERDs.

In order to provide a user friendly interface, a java applet, utilizing the Grappa package was implemented. The interface produced by the applet is shown in Figure 26. Based on initial ERD layouts generated by the Graphviz layout rendering engine (*dot* or *neato*), the java applet provides the following functionalities:

- Create new entities, relations, attributes and connections.
- Delete entities, relations, attributes and connections.
- Rename entities, relations, attributes and connections
- Move any ERD element.
- Zoom in, zoom out and zoom fit.
- Generate PNG graph.
- Generate new DOT files with user modifications.
- Print.

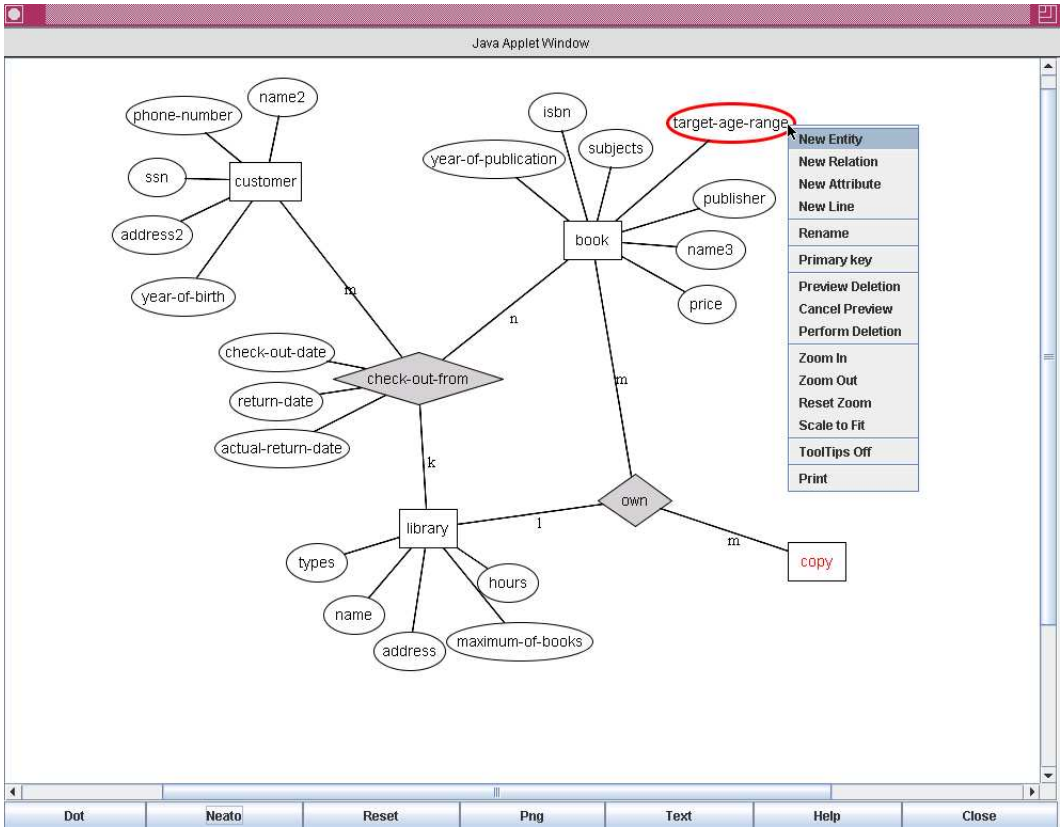


Figure 26: The java applet interface of the ACDM system

4.2 AN EXAMPLE

In this section, a detailed example is provided to illustrate how the system works. Taking the following database requirements specification as the input,

> (1) A concert season schedules one or more concerts. (2) A concert season is identified by its opening date, which includes month, day, and year. (3) A concert includes one or more compositions. (4) A concert is identified by its number. (5) A concert also has a concert date, which consists of month, day, year, and time. (6) For each concert there is one conductor. (7) A conductor is identified by his/her PID. (8) A conductor name also needs to be included in the database. (9) Each composition may require zero or more soloists. (10) A composition is identified by its CID, which consists of composer name and composition name. (11) Compositions have multiple movements. (12) A movement is identified by an MID, which includes movement name and movement number. (13) A soloist is identified by his/her PID. (14) A soloist name is also kept in the database.[5]

First, the preprocessing module performs some substitutions and concatenations as discussed in Section 4.1.2. Then the input text is sentencified. The Attribute Extractor module takes the tagged sentences from the Stanford Tagger with the support of the domain knowledge and WordNet to extract possible attributes. Sentence (2) , (5) , (10) and (12) are covered by the regular expressions and the following attributes are extracted for the corresponding entities or relations.

```
concert-season: opening-date,month,day,year
concert: concert-date,month,day,year,time
composition: cid,composer-name,composition-name
movement: mid, movement-name,movement-number
```

The first element of each list is the entity or relation name followed by the possible attributes. These attribute lists will be merged with the results generated from the heuristic rules later.

All other sentences that are not covered by the Attribute Extractor module are passed to the Stanford Parser. Based on the typed dependency parsing results, a list of *unit* and *td* facts are generated,

[5]For illustration purpose, the sentences are labeled with sentence numbers which are not used in the actual input.

```
(td (td_name det) (word1 "season") (position1 3)
                  (word2 "a") (position2 1) (sent_id 0))
(td (td_name nn) (word1 "season") (position1 3)
                 (word2 "concert") (position2 2) (sent_id 0))
(td (td_name nsubj) (word1 "schedules") (position1 4)
                    (word2 "season") (position2 3) (sent_id 0))
...
(unit (word "a") (root "a") (pos DT) (position 1) (sent_id 0))
(unit (word "concert") (root "concert") (pos NN) (position 2) (sent_id 0))
(unit (word "season") (root "season") (pos NN) (position 3) (sent_id 0))
...
```

With these facts, some of the heuristic rules are fired and the following entity relationship tuples are generated in the format:

$< relation, entity1, entitity2, cardinality1, cardinality2, rule\text{-}name, sentence\text{-}id >$

```
schedule , concert season,  concert, ONE, MANY, a0, 0
include ,  concert,  composition, ONE, MANY, a0, 2
be for,  conductor,  concert, ONE, MANY, e1, 5
require ,  composition,  soloist, MANY, MANY, a0, 7
have ,  composition,  movement, MANY, MANY, a0, 9
```

In addition, the heuristic rules produce a set of entity attributes:

```
concert:  number
conductor:  pid
conductor:  name
soloist:  pid
soloist:  name
```

The ERD Merger module merges these results with the attribute list generated from the Attribute Extractor module. Then the WordNet based entity attribute disambiguation and the Web corpus based entity relation disambiguation are performed. After that, the DOT language generation module generates the following DOT file,

```
graph ER {
 node[shape=box, fontname="cmr10.ttf" ];"concert";"conductor";"soloist";
     "concert-season";"composition";"movement";
 node[shape=ellipse, fontcolor=black, fontname="cmr10.ttf"];"pid2";
     "movement-name";"name";"composition-name";"cid";"mid";"month2";"time";
     "day2";"opening-date";"year2";"month";"concert-date";"name2";"year";
     "number";"composer-name";"movement-number";"day";"pid";
 node[shape=diamond, style=filled, fontcolor=black, color=lightgrey];"require";
     "include";"schedule";"have";"be-for";
```

```
    "composition" -- "require" [label="m"]
    "concert" -- "include" [label="1"]
    "require" -- "soloist" [label="n"]
    "be-for" -- "conductor" [label="1"]
    "concert-season" -- "schedule" [label="1"]
    "composition" -- "include" [label="m"]
    "have" -- "movement" [label="n"]
    "be-for" -- "concert" [label="m"]
    "concert" -- "schedule" [label="m"]
    "composition" -- "have" [label="m"]
    "opening-date" -- "year"
    "cid" -- "composer-name"
    "conductor" -- "pid"
    "mid" -- "movement-name"
    "concert" -- "number"
    "concert-date" -- "year2"
    "concert" -- "concert-date"
    "pid2" -- "soloist"
    "mid" -- "movement-number"
    "month" -- "opening-date"
    "concert-date" -- "day2"
    "day" -- "opening-date"
    "cid" -- "composition-name"
    "mid" -- "movement"
    "concert-date" -- "time"
    "concert-season" -- "opening-date"
    "cid" -- "composition"
    "concert-date" -- "month2"
    "conductor" -- "name"
    "name2" -- "soloist"

  label = "\n\n Engity Relationship Diagram\n";
   fontsize = 14; fontname = "cmr10.ttf"  }
```

An ERD diagram can be generated with this DOT representation using the *dot* or the *neato* render engine. With the Grappa based java applet interface, users can edit and manipulate the ERD diagram in order to produce a better result, especially after taking actions on those elements with low confidence scores. An example result is shown in Figure 27.

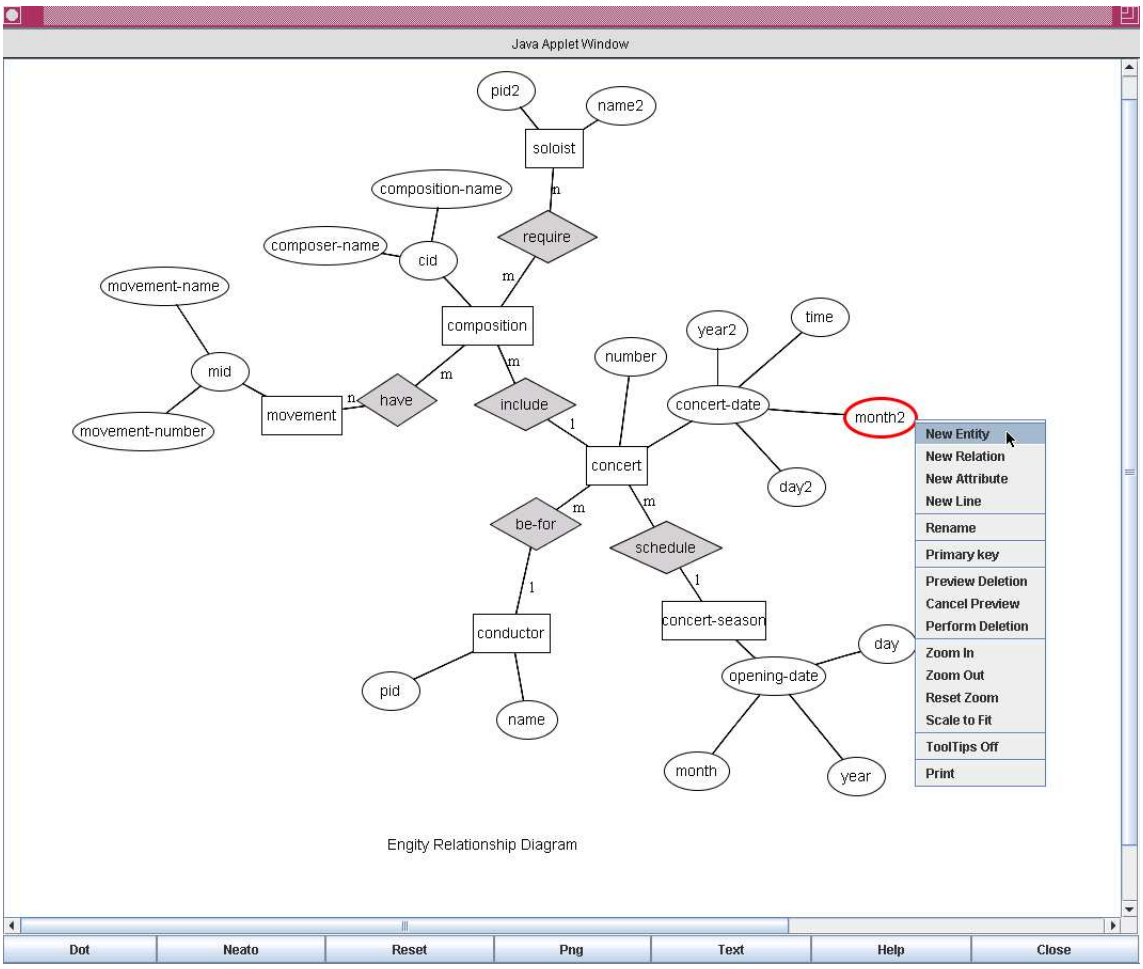


Figure 27: An example ERD generated by the ACDM system

5.0 EVALUATION

5.1 OBJECTIVE

The automated conceptual data modeling tool implemented is not expected to be perfect in transforming natural language database requirements to formal ERDs. This is actually the case for any NLP application because of the inherent informal and ambiguous characteristics of natural language. It is also hard to determine a perfect solution to many of the database conceptual data modeling problems. So one useful benchmark is to determine whether the automated conceptual data modeling tool can help relatively inexperienced people in database design tasks.

The general hypothesis is that it takes less time for human subjects with limited experiences in database conceptual data modeling to work out comparable or even better solutions based on the ERDs produced by the ACDM tool than those without the aid of the ACDM tool. Because both the quality of the ERDs and the time used to produce the ERD are of interest, the above hypothesis essentially contains two sub-hypotheses:

Hypothesis a: It takes less time for human subjects with limited experiences in database conceptual data modeling to work out solutions based on the ERDs produced by the ACDM tool than those without the aid of the ACDM tool.

Hypothesis b: The solutions produced by human subjects with the aid of the ACDM tool are comparable or even better than those produced by human subjects without the aid of the ACDM tool.

It is possible that these hypotheses might hold to different degrees for different levels of complexity of the database requirements problems. For instance, the system might produce less accurate results for more complex problems and these results might be confusing and

less useful for the users. Alternatively, the users might have found the system more useful for the problems they found more difficult. So the experimental design incorporates two levels of problem complexity to provide some sensitivity for these possible outcomes. The complexity levels of the database design problems are assessed based on both quantitative and qualitative criteria. The quantitative criteria refer to the number of elements such as attributes, entities and relations that should be incorporated in the desired ER models. Increasing the number of attributes may not increase the problem complexity levels. However, increasing the number of entities and relations can result in a significant increase in problem complexity. The qualitative criteria refer to the complexity levels of the sentence structures such as coordination conjunctions, compound nouns, pronoun reference, noise input, etc. Human subjects may be more sensitive to the quantitative criteria while the system is more sensitive to the qualitative criteria. Hence, the complexity levels of the problems were judged by a combination of the two criteria.

5.2 EXPERIMENTAL DESIGN

A 2x2 within-subjects (repeated-measures) design was used. The two factors (independent variables) are system (with and without the aid of the ACDM system) and problem complexity levels (easier and harder database design problems). The dependent variables are the time used to produce solutions to database design problems and the quality scores of the solutions. The design is illustrated in Figure 28.

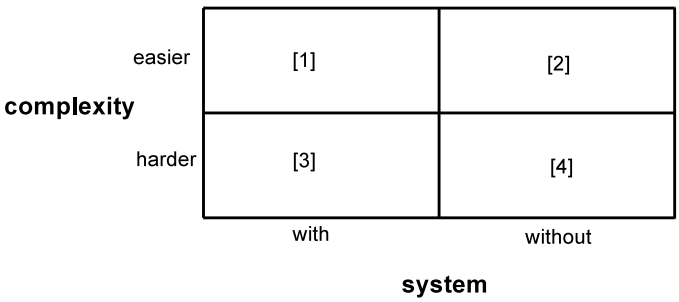


Figure 28: A 2x2 within-subjects design

5.3 EXPERIMENT

A Latin square style arrangement was utilized to balance the learning effects between with and without the ACDM system aid as shown in Table 7. The learning effects between easier and harder problems were not designed to be balanced because the human subjects were allowed to warm up with a easy problem and then work on a relatively harder problem later.

Ten human subjects were divided into five groups. Each of the subjects worked on four database requirements problems: one easier and one harder database design problem with the aid of the ACDM system, one easier and one harder database design problem without the aid of the ACDM system. (The subjects were not told whether the given database design problem was easier or harder). The other subject in the same group worked on the same four database requirements problems but changed the order of with/without the aid of the ACDM system. That is if subject A worked on problem X with the aid of the system, then

Table 7: Latin Square design to balance learning effects

Subject	Problem1	Problem2	Problem2	Problem4
s1	e1/wo	e2/w	h1/wo	h2/w
s2	e1/w	e2/wo	h1/w	h2/wo
s3	e3/wo	e4/w	h3/wo	h4/w
s4	e3/w	e4/wo	h3/w	h4/wo
s5	e5/wo	e6/w	h5/wo	h6/w
s6	e5/w	e6/wo	h5/w	h6/wo
s7	e7/wo	e8/w	h7/wo	h8/w
s8	e7/w	e8/wo	h7/w	h8/wo
s9	e9/wo	e10/w	h9/wo	h10/w
s10	e9/w	e10/wo	h9/w	h10/wo

subject B would work on problem X without the aid of the system. Hence, a total of ten easier and ten harder database design problems were used (see Appendix G). Each of the database design problems was done by two subjects, one with the aid of the ACDM system and one without the aid of the ACDM system.

The Gnome Dia program was used by subjects to construct ERDs without the aid of the ACDM system. Compared with the user interface of the ACDM tool, the Dia program is more powerful and easy to use. Although none of the subjects had used the Gnome Dia program before, after a short introduction of the Dia program, all of the subjects were comfortable using it. Thus any difference between the with and without system aid groups due to interface differences would be conservative in that the without the system aid group had the better interface.

Each subject could take as long as he/she wanted to construct an ERD for each of the database design problems.

5.4 ANALYSIS OF EXPERIMENTAL RESULTS

5.4.1 Subjects

All of the subjects were students in the Department of Information Science and Telecommunications at the School of Information Sciences, University of Pittsburgh. Two were undergraduates and eight were graduate students.

Four subjects did ERD exercises in database design courses some years ago while the other six constructed some small ERDs for a class assignment or project recently.

Seven subjects had used the Microsoft Visio program and one had used Microsoft Paint to construct ERDs while the other two only drew ERDs on paper previously.

5.4.2 Obtaining experimental data

The quality of the ERDs produced by the subjects was judged by a third party, a Ph.D student who had been a teaching assistant for a graduate database design course for several

terms. The judge was not told whether the ERDs were produced by subjects with or without the aid of an automated conceptual data modeling tool. The ERD pairs were displayed to the judge as shown in Figure 29.

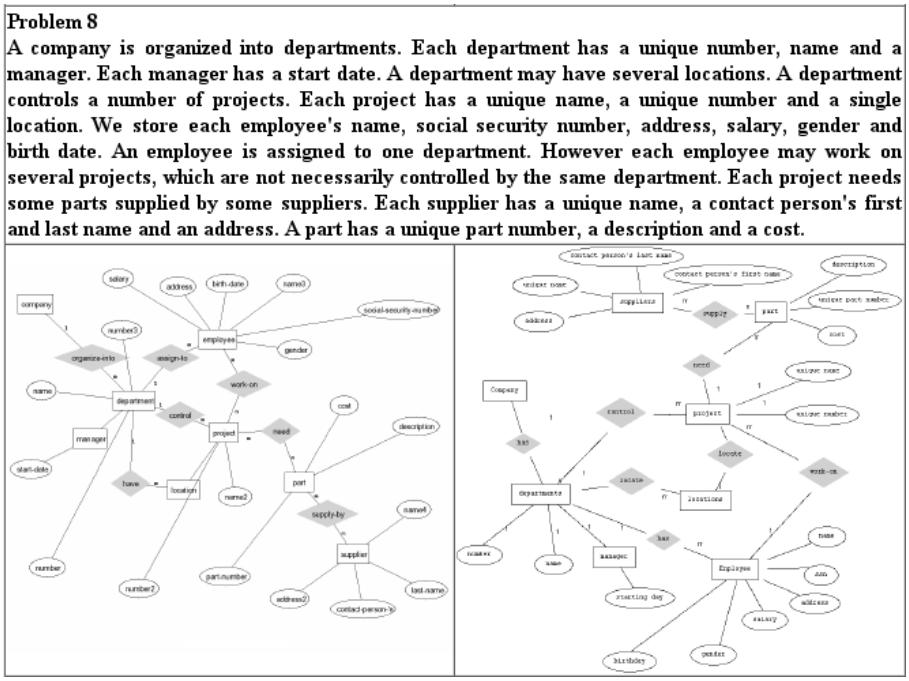


Figure 29: An ERD quality judgment scenario

The following guideline was used to assess the quality of the ERDs.

- Attribute (1 point each)
  - Add 1 point for each correct attribute stated in the requirements.
  - No penalty for very likely attributes of an Entity but not stated in the requirements such as “SSN” for “employee”.
  - Deduct 0.5 point for each wrong attribute.
  - Add 1 point for each subcomponent of a compound attribute.
  - No penalty for reasonable paraphrase of the name, such as “concert number” → “concert id”.

- Entity (2 points each)
  - Add 2 points for each correct entity stated in the requirements.
  - No penalty for very likely entity but not stated in the requirements.
  - Deduct 1 point for each wrong entity.
  - Do not count compound attributes as entities.
- Relation (3 points each)
  - Add 3 points for each relation that is correctly attached to the corresponding entities.
  - A ternary relation is equal to two binary relations, hence carries 6 points.
  - No penalty for very likely relations added by subjects.
  - Deduct 1.5 point for each wrong relation.
- Cardinality (total 3 points)
- Overall diagram quality (total 5 points)
  - Readability of the attribute, entity, relationship names.
  - Alignment and arrangement of diagram elements.

Based on this assessment guideline, the ERD quality scores are shown in Table 8.

The time used by each subject to construct the ERDs is shown in Table 9

Table 8: ERD quality scores

subject	complexity	system	quality	subject	complexity	system	quality
s1	e	w	36	s6	e	w	52
s1	e	wo	37	s6	e	wo	39
s1	h	w	42.5	s6	h	w	50
s1	h	wo	30	s6	h	wo	47
s2	e	w	51	s7	e	w	46
s2	e	wo	33	s7	e	wo	28
s2	h	w	60	s7	h	w	61
s2	h	wo	42	s7	h	wo	28
s3	e	w	35	s8	e	w	43
s3	e	wo	56	s8	e	wo	30
s3	h	w	61.5	s8	h	w	85
s3	h	wo	46	s8	h	wo	45.5
s4	e	w	52	s9	e	w	31
s4	e	wo	22.5	s9	e	wo	23
s4	h	w	56.5	s9	h	w	35.5
s4	h	wo	66	s9	h	wo	16
s5	e	w	36	s10	e	w	32
s5	e	wo	24	s10	e	wo	32
s5	h	w	42	s10	h	w	78
s5	h	wo	38	s10	h	wo	38

Table 9: Time used to construct ERDs

subject	complexity	system	time	subject	complexity	system	time
s1	e	w	13	s6	e	w	13
s1	e	wo	26	s6	e	wo	22
s1	h	w	9	s6	h	w	14
s1	h	wo	25	s6	h	wo	18
s2	e	w	11	s7	e	w	9
s2	e	wo	17	s7	e	wo	16
s2	h	w	31	s7	h	w	12
s2	h	wo	27	s7	h	wo	17
s3	e	w	15	s8	e	w	9
s3	e	wo	42	s8	e	wo	21
s3	h	w	15	s8	h	w	26
s3	h	wo	33	s8	h	wo	37
s4	e	w	17	s9	e	w	17
s4	e	wo	13	s9	e	wo	30
s4	h	w	13	s9	h	w	13
s4	h	wo	17	s9	h	wo	30
s5	e	w	6	s10	e	w	5
s5	e	wo	15	s10	e	wo	12
s5	h	w	9	s10	h	w	15
s5	h	wo	15	s10	h	wo	10

5.4.3 ANOVA analysis of the quality of the ERDs

Table 10: ANOVA analysis of ERD quality

Source	F value	Pr(>F)
Main effect: system	20.644	0.0014 **
Main effect: complexity	11.103	0.0088 **
Interaction: complexity $*$ system	1.300	0.2836

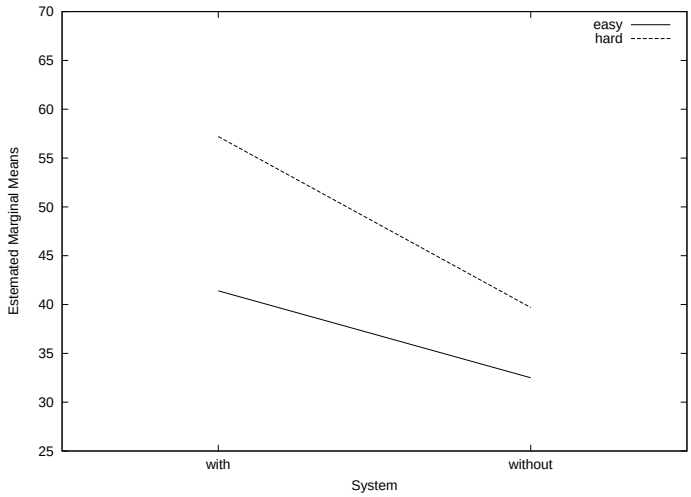


Figure 30: Mean quality scores as a function of problem complexity levels and system

A 2x2 within-subjects analysis of variance was performed on ERD quality scores as a function of ACDM system (with, without) and problem complexity levels (easier, harder), as shown in Table 10.

The main effect of with/without system was supported (null hypothesis was rejected at $p = .0014$) supporting the hypothesis that the system helped subjects produce better ERDs than they did without it.

Tha lack of interaction suggests that the effect of the with/without main effect was not significantly different at the two levels of problem complexity.

The more difficult problems produced significantly higher scores than the easier ones but that could be an artifact of the scoring method and was not central to the hypotheses of interest.

5.4.4 ANOVA analysis of the time used to construct ERDs

Table 11: ANOVA analysis of the time used to construct ERDs

Source	F value	Pr(>F)
Main effect: system	13.729	0.0049 **
Main effect: complexity	1.552	0.2443
Interaction: complexity ∗ system	1.7261	0.2214

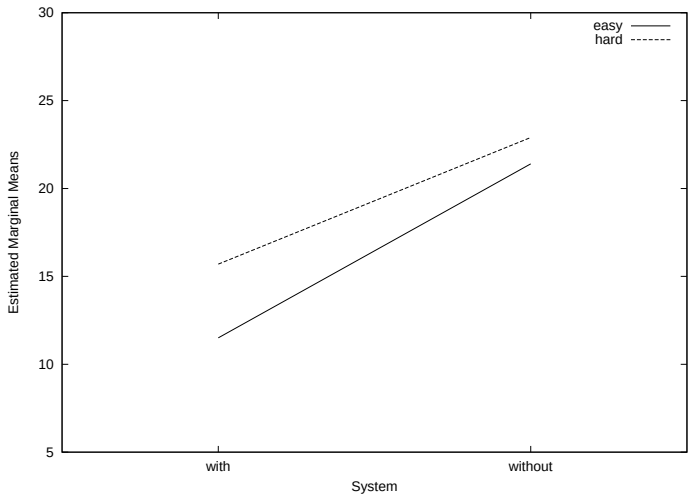


Figure 31: Mean time as a function of problem complexity levels and system

Similar to the analysis of the ERD quality scores, a 2x2 within-subjects analysis of variance was performed on the time used to construct ERDs as a function of ACDM system aid (with, without) and problem complexity levels (easier, harder), as shown in Table 11.

The main effect of with/without system was supported (null hypothesis was rejected at $p = .0049$) supporting the hypothesis that the system helped the subjects produce ERDs faster than they did without it.

The lack of interaction suggests that the effect of the with/without main effect was not significantly different at the two levels of problem complexity.

The time used to construct ERDs for the more difficult problems was not significantly higher than that for the easier problems.

Based on the ANOVA analysis of the quality scores and the time used to construct ERDs, the hypothesis that it takes less time for human subjects with limited experiences in database conceptual data modeling to work out comparable or even better solutions based on the ERDs produced by the ACDM tool than those without the aid of the ACDM tool is supported.

5.4.5 Post-experiment questionnaire

Table 12: Summarizing how subjects felt about the problem complexity

	Easy1 (w, o)	Easy2 (w, o)	Hard1 (w, o)	Hard2 (w, o)
a) very easy	2 (2, 0)	1 (1, 0)	1 (1, 0)	2 (1, 1)
b) somewhat easy	6 (1, 5)	7 (3, 4)	1 (0, 1)	3 (2, 1)
c) somewhat hard	2 (2, 0)	2 (1, 1)	4 (3, 1)	5 (2, 3)
d) very hard	0 (0, 0)	0 (0, 0)	4 (1, 3)	0 (0, 0)

The subjects were asked to fill out a post-experiment questionnaire (see Appendix I). Table 12 summarizes how subject felt about the problem complexity. For instance, the numbers in the first cell "2 (2, 0)" represent that a total of two subjects felt the first problem was very easy. The numbers inside the parenthesis are the number of subjects from the group using the ACDM system and that from the group without using the system respectively. In this case, these two subjects were from the group using the system. No subject from group without using the system felt the first problem very easy. In general, subject's judgments

Table 13: Summarizing subjects' confidence to their solutions

	Easy1 (w, o)	Easy2 (w, o)	Hard1 (w, o)	Hard2 (w, o)
a) not confident	0 (0, 0)	0 (0, 0)	3 (1, 2)	0 (0, 0)
b) somewhat not confident	2 (0, 2)	0 (0, 0)	1 (1, 0)	1 (0, 1)
c) somewhat confident	3 (2, 1)	7 (3, 4)	3 (1, 2)	4 (2, 2)
d) very confident	5 (3, 2)	3 (2, 1)	3 (2, 1)	5 (3, 2)

were consistent with the pre-designed problem complexity levels. Furthermore, there is a suggestion that subjects felt the problems were easier when they worked with the ACDM tool aid than without the system aid[1].

Table 13 summarizes how confident the subjects felt about their solutions to the database design problems. In general, the subjects were more confident about their solutions to the easier database design problems than those to the harder database design problems; the subjects with the ACDM tool aid felt more confident about their solutions than those without the system aid[1].

The subjects were asked for their opinions of the advantages and disadvantages of using the ACDM tool:

- The major advantages of using the ACDM tool are:
  - The system extracts attributes and entities accurately and prevents users from overlooking some attributes;
  - The system provides users with an initial model to work on which saves some time;
  - The system is very helpful and efficient to construct not very complicated ERDs.
  - Users can focus more on the database design problems instead of the drawing process.
- The major disadvantages of using the ACDM tool are:
  - The relations extracted were not very accurate;
  - If the initial ERDs were wrong, it is not easy to fix.
  - Some often used functionalities such as copy, cut, past, are missing;

[1]based only on slight evidence from the table, not on statistical analysis.

Most of the subjects thought the ACDM tool was very helpful and did not prevent them from working efficiently. Some subjects thought the ACDM was cool and would like to use it for database design courses.

5.4.6 A Precision-Recall analysis of the experiment results

Table 14: Precision and Recall evaluation

	User		System		User+System	
	Precision	Recall	Precision	Recall	Precision	Recall
Attribute (A)	88.4% [1]	98.4%	94.3%	93.2%	93.3%	96.0%
Entity (E)	95.1%	61.9%	82.6%	90.4%	93.8%	96.0%
Relation (R)	88.4%	66.1%	80.6%	88.2%	90.9%	94.5%
Overall	89.6 %	89.9%	87.5%	84.7%	92.8%	92.4%

Some of the previous research projects have adopted *precision* and *recall* to evaluate their systems (see section 2.2.2). Although it is still questionable and inconsistent to utilize precision and recall to evaluate the performance of automated conceptual data modeling systems, especially when there is no standard evaluation requirements corpus available, these measures do provide some comparisons to previous work and indications of overall quantitative performance. Hence the precision and recall are reported here.

The precision and recall were calculated for each of the three ERD element categories: Attribute, Entity and Relationship using the formulae (2.2) and (2.3) for the ERDs generated by the human subjects, the ACDM system alone and the human subjects with ACDM system.

$$Recall = \frac{N_{correct}}{N_{correct} + N_{missing}} \quad \text{[OHM04]} \qquad (2.2)$$

$$Precision = \frac{N_{correct}}{N_{correct} + N_{incorrect}} \quad \text{[HG03]} \qquad (2.3)$$

[1] This number was lowered significantly by one human subject who added many reasonable attributes such as id for each entity. However, these attributes were not specified in the requirements.

$N_{correct}$ is the number of correct items (attributes, entities or relations) in the target evaluation set. $N_{missing}$ is the number of desired items that are not extracted in the target evaluation set. $N_{incorrect}$ is the number of incorrect items in the target evaluation set. The target evaluation sets are ERDs generated by the human subjects, ERDs generated by the ACDM system alone and ERDs generated by the human subjects with the ACDM system.

As shown in table 14, human subjects (Users) preformed relatively well on entity and relation extraction in terms of precision but poor in terms of recall while the system performed relatively well on attribute extraction in terms of recall. Because of this complementary effects, the human subjects with the ACDM system group (User+System) has the highest overall scores in both precision and recall.

5.5 LIMITATIONS OF SYSTEM COMPONENTS

The overall system performance has been evaluated in previous section. In this section, the limitations of some of the system components will be discussed.

One character of the ACDM system is integrating multiple modules and resources for the purpose automated conceptual data modeling. Some of the modules are specifically designed for this application while others are adopted from open source third party packages. Each component has some limitations.

The system lacks reasoning capabilities to handle complicated cases. The regular expressions that are used to handle long enumerative attribute sentences work well on most of the cases and can extract the attributes correctly with proper names. However, there are some exceptions and difficulties. For instance, in the requirements statement "discount type, purchase quantity and actual price should be stored in database.", the system could not extract the attributes "discount type", "purchase quantity" and "actual price". Even if the system could extract these three attributes with some additional regular expressions, it is very challenging to attach them to the correct entity or relation. The requirements specification does not explicitly state which entity these three attributes should attach to. It may be obvious to human beings that these three attribute should attach to a "purchase"

relation. However, the system do not possess this kind of reasoning capabilities. The heuristic rule based ER Extractor can only extract most of the desired entities and relations that are explicitly stated in the requirements documents but not those that need further reasoning. For instance, in the statement "each employee has a manager", if the statement "a manager is also an employee" is not specified in the requirements documents, the system doesn't reason about it. However, this problem can be addressed to some extent by utilizing WordNet or other ontology resources.

The heuristic rules proposed are not complete. There are certain sentence structures that are not covered. Some of the sentence structures are too general to enable the extraction of some specific entities and relations such as in the statement "computer manufacturing community provides sophisticated and complete computer configurations by outsourcing services from several trading partners". Some of the sentence structures involve advanced reference resolution which the system does not possess. However, the system relies on two mechanisms to reduce the impact of this problem. One is over-generation. The heuristic rules proposed in section 3.4 will over-generate many entity relationship tuples while the heuristic rules proposed in section 3.5 refine and filter out the inappropriate entity relationship tuples. For instance, without the refining rules, the system over-generated 28.8% of the tuples for the example problems used in the evaluation experiment. The other mechanism is to take advantages of the redundant information in the requirements. For instance, in statements "A work task is associated with one of 20 different job types. During a given day an employee may perform more than one work task, each associated with a different job type.", the system can extract $\langle$*associate-with work-task job-type*$\rangle$ from the first sentence even if the system fails to resolve the reference of "each" in the second sentence.

Some of the special issues on utilizing the heuristic rules discussed in section 3.5 were not yet addressed well in the ACDM system. Although some kind of remedies have been proposed to address the specific issues, problems exist. For instance, the system only implemented base form based deduplication. The more advanced deduplication mechanisms based on synonym, antonym, abbreviation and paraphrase, which are much more challenging, are not implemented. Therefore some semantically duplicated entities or relations such as "worker union" vs. "union" are produced in the final conceptual data

models by the system. Furthermore, the ternary relation rule (3.27) fails in some cases such as coordination conjunctions sentences. For instance, in the sentence "a bookstore sells books and software to students", three binary relation tuples will be extracted: $\langle sell\ bookstore\ book \rangle$, $\langle sell\ bookstore\ software \rangle$, $\langle sell\text{-}to\ bookstore\ student\ \rangle$. Currently, the system only combines $\langle sell\ bookstore\ book \rangle$ and $\langle sell\text{-}to\ bookstore\ student\ \rangle$ with rule (3.27) and generates $\langle sell\text{-}to\ bookstore\ book\ student \rangle$ and leaves $\langle sell\ bookstore\ software \rangle$ as a binary relation. Moreover, those coordinating conjunctions rules are also problematic in some cases. Rules (3.20 – 3.24) aim at addressing conjunction in the subject or object position. Conjunction of verbs and verbs with objects are not addressed. For instance, in the sentence "each student has only a single major and associated with a single department", the tuple $\langle associte\text{-}with\ student\ department \rangle$ can not be extracted with rule (3.20 – 3.24).

The lexical filter used by the system is not effective in some cases. The use of a general lexical knowledge resource, WordNet, for automated disambiguation tasks in conceptual data modeling is novel and it is quite effective in some of the tasks such as handing "of-structures". However, WordNet does not work well in some cases to distinguish attribute nouns from entity nouns. For instance, in the statement "programming language has name and platform", "platform" is treated as an entity noun according to WordNet. It is problematic to use WordNet in this case because WordNet is a general knowledge resource and is not specific to conceptual data modeling applications. Moreover, the distinction between entity nouns and attribute nouns is fuzzy in some cases.

The utilization of the Web corpus and search facilities as a semantic filter for automated conceptual data modeling is also problematic in some cases. Even with the huge data size of the Web corpus, sparse data is still a problem and causes false negative alarms to filter appropriate entity relationships. In order to reduce the problem, the system does not remove the dubious relations automatically but only flags them with different colors and prompts users to take actions on those cases.

6.0 CONCLUSIONS AND FUTURE WORK

6.1 CONTRIBUTIONS AND CONCLUSIONS

This research aimed at investigating various formalism translation and extraction problems and proposing novel solutions employing the most recent NLP-related technologies and large scale knowledge resources for their specific applications in automated conceptual data modeling. A working system ACDM was developed to demonstrate the feasibility of the suggested solutions to automated ERD construction. To be specific, the contributions of this research are:

- Proposed a relatively complete set of heuristic-rules for automated conceptual data modeling based on high level grammatical relations extracted from the parsing results of state of the art NLP parsers. Compared with the heuristic rules used in previous research in the literature, these rules are based on high level grammatical relations which are more accurate and fine-grained than word classes or other direct patterns. These rules are also more systematically structured and operational.
- Developed an automated database conceptual data modeling tool. The tool incorporated many core modules built specifically for this application and supporting modules adopted from open source packages.
- Demonstrated the utility of the automated database conceptual data modeling system by a human subjects experiment. It was shown that it took less time for human subjects with limited experiences in database conceptual data modeling to work out comparable or even better solutions based on the ERDs produced by the ACDM tool than they did without using the ACDM tool.

- Demonstrated that it is feasible to develop a NL translation tool for specialized NL documents using currently available semantic resources, a NLP parser, heuristic rules and domain specific knowledge.

Various issues regarding the use of semantic and linguistic resources were explored and the reasons for the decisions taken were discussed.

6.2 FUTURE WORK

There are some NLP-related issues that need to be further investigated in the future. The heuristic rules proposed in this research are based on high level grammatical relations from two of the state of the art parsers. The rules are tightly coupled with the taggers used by the specific parsers. In the future, how to decouple the heuristic rules from the taggers used by the NLP parsers will be investigated. Furthermore, the several large scale knowledge resource based semantic filtering mechanisms for entity relationship disambiguation only work to some extent. None of them is perfect for the application of automated conceptual data modeling. More systematic analysis and evaluation of the large scale knowledge resources for semantic analysis needs to be done in the future. Other problems including that of unstated common sense requirements may also need to be addressed in the future.

Although the experimental results are encouraging, more experiments are needed to further assess the performance of the system on more complex requirements documents. Furthermore, requirements validation via reverse natural language generation based on the target formalisms can serve as another way to evaluate the system. Some of the previous systems [RP92, BvdR96] have explored similar approaches. Although it is still quite challenging to generate natural language requirements as natural as the original input requirements, some kind of human readable texts can be generated from underlying formalisms using state of the art natural language generation technologies and resources.

Moreover, the approaches explored in this research may be extended in many other areas with similar requirements such as information extraction in financial reports [Cos97], medical reports [SFL87] and accident insurance records [NDE03] because the central problem for automated conceptual data modeling is formalism extraction.

APPENDIX A

A CHRONOLOGY OF NLP-CDM SYSTEMS

Authors & Year	**Brief**
[Che83]	Pioneering study the corresponding patterns between English sentence structures and the elements of ERDs for heuristic rule based approaches. 11 heuristic translation rules between English sentence structures and ERDs are proposed with detailed examples. However, no system implementation.
[Abb83]	A technique is presented to derive programming data type from common nouns, variables from direct references, operators from verbs and attributes, and control structures from their English equivalents.
[BGM85]	SECSI expert system, 3 input forms: controlled languages, formal language, graph; The NL interface uses very restricted, only subject-verb-complement structured sentences. The system is extensible in the sense that it also offers an interactive interface which allows the database design expert to modify or add design rules to the system dynamically.
[EL85]	Aims at extract the terminological knowledge from database design specifications. The terminological knowledge is put into an S-diagram data structure, which can be used to specify classes (entities) and attributes. The S-diagram is only suitable for small systems.

[Bla87]	Reports an approach with a simple syntactic parser and a semantic analyzer using McCords Slot Grammar. It claims that a simple ERD could be built by examining the attributes of the relational database predicates in the parse tree. However, the main purpose of this paper is to outline an application area and discuss the syntactic, semantic and real-world knowledge needed in conceptual data modeling.
[SHT+87, SHE89]	Verb-oriented process to derive incrementally a formal specification using rule-based methods. Nouns correspond to objects or classes, and verbs correspond to messages. The subject of a sentence corresponds to a sender of the message denoted by its verb. One of its objective words is considered as the receiver of the message. The categorization of nouns and verbs are performed by human interactively.
[RP92, RBA98, RP00]	Linguistic based CASE tool for requirement engineering; A set of patterns that combine cases and classes of verbs are defined. The conceptual schema generation is grounded upon rules that map cases onto concepts. One of the characteristics of this paper is that linguistic approaches have been extensively used to analyze the NL documents. Mainly techniques, syntactic parsing + semantic net representation, pattern match. French.
[TB93]	Reports DMG, a rule-based design tool to translate natural language descriptions to EER structures from parsing results. A large number of heuristic rules (most of them are dealing with the syntactic structure of natural language input) are proposed, however, this tool has not yet developed into practical system. German.
[PAC+93]	Presents a combined language and reasoning engine – CLEAR. CLEAR expresses sentences in Quasi-Logical Form representation, handling reference resolution, compound nominals, and ellipses. This system collects contextual information for the reference resolution and dialog-based interaction. Reasoning is the main focus of the post-processing of NLP.

[FGR+94]	NL2ACTL is designed for the formalization of behavioral requirements in the design of reactive systems. The translation process is specifically designed to work between restricted natural language and ACTL, and it is not based on a general semantic theory of natural language. Direct correspondences between natural language structures (extended forms) and ACTL syntax is identified, without intermediate steps such as building semantic interpretation.
[BD94, BCD+95, BDT97]	A dialog tool, part of a larger database design system (RADD) that includes a syntax analyzer, a semantic role definer and a pragmatic interpreter, is reported. The aim of this tool is to elicit the structure, semantics and behavior of a database from a user, then build conceptual EER models from the requirements. They distinguish three types of questions, content questions, linguistic clarification questions, and pragmatic clarification questions. German.
[NR95]	Focuses on the requirements validation problems in Object Oriented Analysis (OOA) by applying the Link Parser to a requirements document, extracting candidate objects, methods and associations, composing them into an object model diagram, and then comparing the results to those determined by manual OOA. No sophisticated analysis of any of the NLP hard problems, such as parser inadequacy, ambiguity, and domain knowledge representation.
[Mic96]	A CASE tool Natural Language Object Oriented Production System (NL-OOPS) that supports requirements analysis by building object oriented models from natural language requirements. An NLU subsystem, Large-scale Object-based Linguistic Interactor Translator Analyzer (LOLITA) is used for full natural language analysis. An algorithm that extracts objects and their associations for model building is discussed.

[BvdR96]	A natural language and scenario-based approach to requirement engineering. Formal event models (CEMs) are used to specify dynamic information of a scenario in a natural way. CEMs are created during a particular human-computer interaction assisted with WordNet. The event models are also translated into analysis and design models on object level, which bridges the gap between informal requirement engineering and object-oriented analysis and design. German.
[OM96]	A system (Newspeak) is presented for a controlled language. Logical forms are used for semantic representation. The system focuses on ambiguity detection and resolution. The system can not resolve anaphoric references, and does not address missing words problem.
[FKM+96, FKM+98]	Focuses on determining the cardinalities of relationships and disambiguating part-of relationships by examining cardinality designators. A specific linguistic approach called NTS and a particular conceptual modeling approach, the Klagenfurt Conceptual Predesign Model (KCPM) is used in this approach.
[Mor97]	Aims at formalizing the Object-Oriented analysis by the linguistic pattern in the requirements. A comprehensive method that includes nine stages such as identifying synonyms, static requirements structuring, dynamic requirements structuring and object model construction. Spanish.
[AG97, AG06]	Circle, a web-based comprehensive environment for aiding in natural language requirements gathering, elicitation, selection, and validation. Human generated glossary and minimal domain description are added to the original requirements. The actual recognition is performed by a number of MAS (Model, Action, Substitution) rules. A detailed case study of a fictitious missile control system is provided and various stages of requirements analysis are covered.

[Fou99]	The processing of the language breaks down into two stages: acquisition of domain knowledge, and linguistic analysis. Linguistic analysis sub-divided into five stages of analysis: morphological, lexical, syntactic, semantic, and pragmatic. After language processing, each proposition is represented by a concept graph. Rules offer the possibility to join or disassociate different concepts. The final specifications are represented in Z language, a formal language used for software specifications.
[GSD99]	Rule-based ER generator that consists of two major components: a natural language understanding (NLU) system and an ER-generator. Different from other approaches, the ER-generator takes a specific final knowledge representation structure as input, which is essentially a frame-like structure with values filled by the semantic interpreter. The ER-generator is a rule-based system that utilizes two kinds of rules: specific rules that link to the semantics of some words in the sentences, and generic rules that identify ER-entities and ER-relationships on the basis of the logical form of the sentence
[LB02a, LB02b, LB02c, Lee03]	An automated conversion from NL to an executable formal specification using two-level grammar (TLG) and contextual natural language processing. A knowledge-base is built from the NL documents via a middle representation (XML), which stores the syntactic, semantic and pragmatic information of the requirements. The meta-attribute information from XML is also stored in the knowledge base. The final formal representation is VDM++.
[Gal02]	The corresponding patterns between the basic elements of NL and basic constituents of different models, like objects, process, dataflow and workflow, have been analyzed. The first goal of this paper is to summarize the main features of natural language and their counterparts in conceptual data modeling. The second goal of this paper is to sketch a possible solution to the integration of the conceptual models.

[HG03]	An NL-based CASE tool called Class Model Builder (CM-Builder) aimed at supporting the conceptual analysis stage in software development for an object-oriented framework. CM-Builder can work either automatically or interactively with an analyst. UML class model is produced as the target formalism. It also used discourse interpretation and frequency analysis in producing the conceptual models.
[SC04]	A requirements assessment architecture that combines natural language parsing and artificial intelligence. A prototype system (REWARD) is partially implemented. An interesting observation is that there is definite correlation between the ability of REWARD to parse a requirement and the ability of humans to understand the writers intention.
[OHM04]	Heuristic-based ER-converter. The heuristic rules are associated with weights according to the confidence level that the event is true. The weights assigned to each rule are mainly based on intuition. It does provide secondary flexibility to control the rule to be fired or not in the processing. User interventions are required when the calculated weights are low in the processing.
[DM06]	An automated multi-component approach. The system is a fully integrated composite of existing, public available components including a parser (Link Parser), a lexical filter (WordNet) and a semantic filter (Google web corpus search facilities). Good performance on precision and recall compared with the state-of-the-art.

APPENDIX B

FREQUENCY OF SCOPE WORDS

Scope words	**Requirements**		**TDT2**	
	Counts	Percent	Counts	Percent
a	826	4.82%	489,964	2.41%
an	173	1.01%	77,490	0.38%
one	161	0.94%	50,436	0.25%
two	21	0.12%	35,032	0.17%
each	341	1.99%	8,138	0.04%
every	37	0.22%	8,079	0.04%
some	38	0.22%	35,034	0.17%
several	60	0.35%	9,486	0.05%
multiple	18	0.10%	425	0.002%
any	44	0.26%	17,547	0.09%
many	35	0.20%	22,359	0.11%
all	39	0.23%	37,417	0.18%
no	115	0.67%	32,621	0.16%
not	61	0.36%	79,931	0.39%
never	2	0.01%	8,841	0.04%
Total	**1971**	**11.50%**	**912,530**	**4.50%**

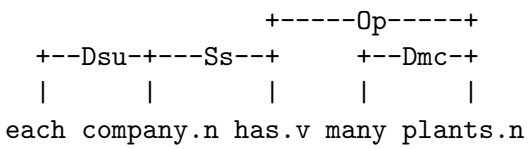
Note: The TDT2 English Corpus includes 6 news sources from January 4 to June 30, 1998.

APPENDIX C

LINK TYPE BASED HEURISTIC RULES

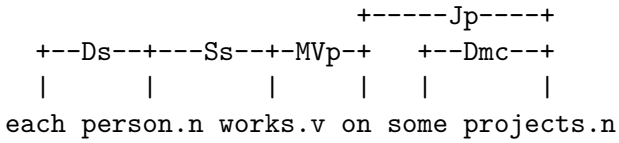
$$S[sp]^* + O[sp]^* \;\Rightarrow\; \langle S.RW \quad S.LW \quad O.RW \rangle \tag{C.1}$$

For instance,



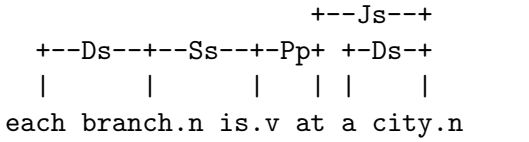
$$S[sp]^* + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle S.RW\text{-}MV.RW \quad S.LW \quad J.RW \rangle \tag{C.2}$$

For instance,



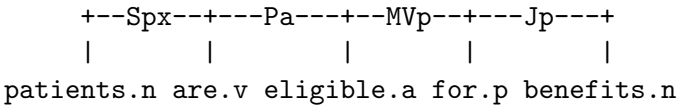
$$S[sp]^* + P[sp]^* + J[sp]^* \;\Rightarrow\; \langle S.RW\text{-}P.RW \quad S.LW \quad J.RW \rangle \tag{C.3}$$

For instance,



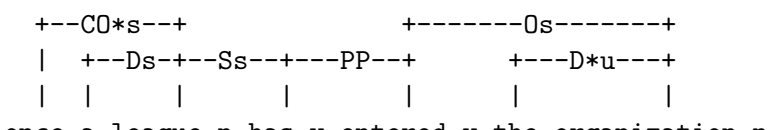
$$S[spx]^* + Pa + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle Pa.RW \quad S.LW \quad J.RW \rangle \tag{C.4}$$

For instance,



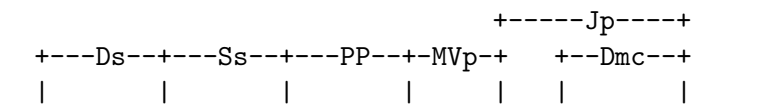
$$S[sp]^* + PP + O[sp]^* \;\Rightarrow\; \langle PP.RW \quad S.LW \quad O.RW \rangle \tag{C.5}$$

For instance,



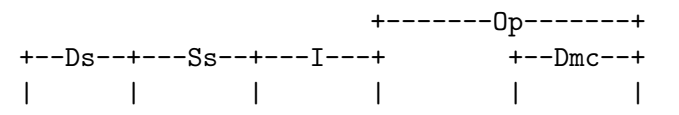
$$S[sp]^* + PP + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle PP.RW\text{-}MV.RW \quad S.LW \quad J.RW \rangle \tag{C.6}$$

For instance,



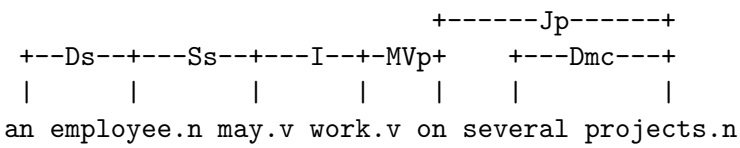
$$S[sp]^* + I + O[sp]^* \;\Rightarrow\; \langle I.RW \; S.LW \; O.RW \rangle \tag{C.7}$$

For instance,



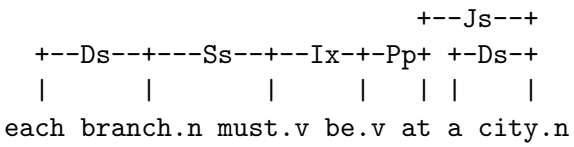
$$S[sp]^* + I + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle I.RW\text{-}MV.RW \quad S.LW \quad J.RW \rangle \tag{C.8}$$

For instance,



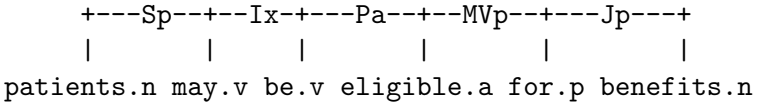
$$S[sp]^* + I[x]^* + P[p]^* + J[sp]^* \;\Rightarrow\; \langle I.RW\text{-}P.RW \quad S.LW \quad J.RW \rangle \tag{C.9}$$

For instance,



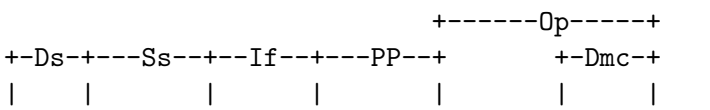
$$S[sp]^* + I[x]^* + Pa + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle Pa.RW \quad S.LW \quad J.RW \rangle \tag{C.10}$$

For instance,



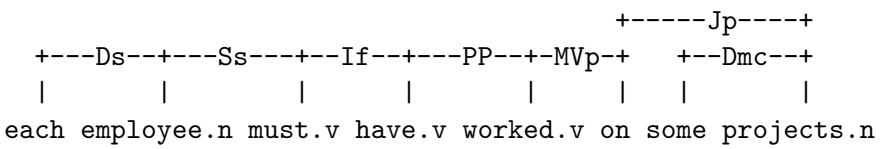
$$S[sp]^* + I[f]^* + PP + O[sp]^* \;\Rightarrow\; \langle PP.RW \quad S.LW \quad O.RW \rangle \tag{C.11}$$

For instance,



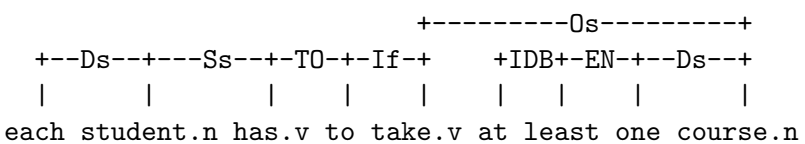
$$S[sp]^* + I[f]^* + PP + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle PP.RW\text{-}MV.RW \quad S.LW \quad J.RW \rangle \tag{C.12}$$

For instance,



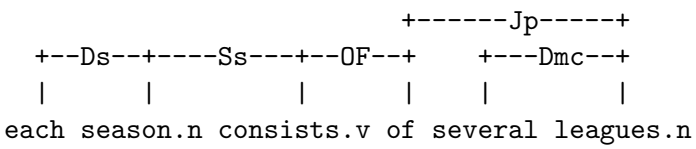
$$S[sp]^* + TO + I + O[sp]^* \;\Rightarrow\; \langle I.RW \quad S.LW \quad O.RW \rangle \tag{C.13}$$

For instance,



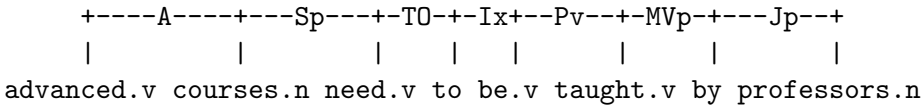
$$S[sp]^* + OF + J[sp]^* \;\Rightarrow\; \langle S.RW\text{-}OF.RW \quad S.LW \quad J.RW \rangle \tag{C.14}$$

For instance,



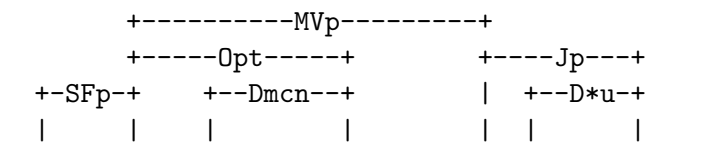
$$S[sp]^* + TO + I[x]^* + Pv + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle Pv.RW \quad S.LW \quad J.RW \rangle \tag{C.15}$$

For instance,



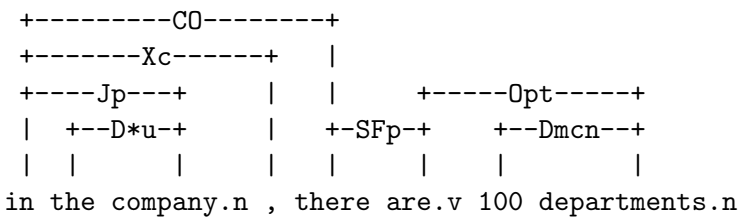
$$SF[sp]^* + O[spt]^* + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle SF.RW\text{-}MV.RW \quad O.RW \quad J.RW \rangle \tag{C.16}$$

For instance,



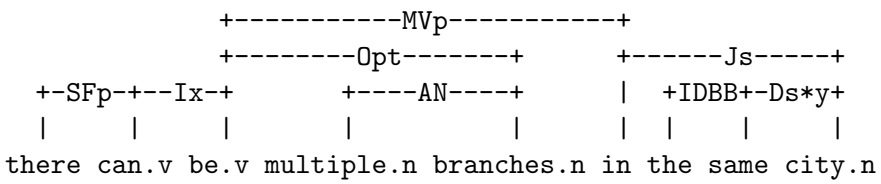
$$J[sp]^* + CO + SF[sp]^* + O[spt]^* \;\Rightarrow\; \langle SF.RW\text{-}CO.LW \quad O.RW \quad J.RW \rangle \tag{C.17}$$

For instance,



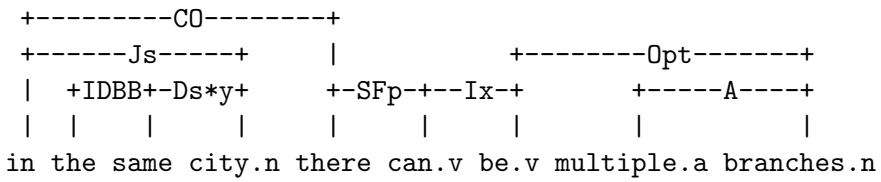
$$SF[sp]^* + I[x]^* + O[spt]^* + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle I.RW\text{-}MV.RW \quad O.RW \quad J.RW \rangle \quad \text{(C.18)}$$

For instance,



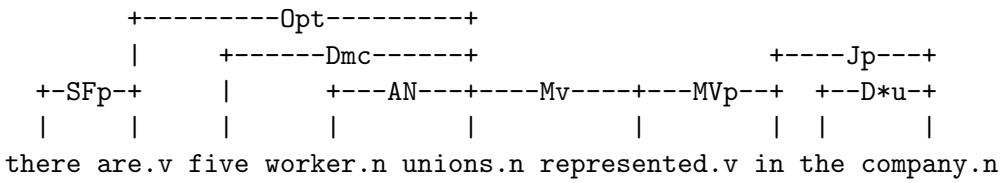
$$J[sp]^* + CO + SF[sp]^* + I[x]^* + O[spt]^* \;\Rightarrow\; \langle I.RW\text{-}CO.LW \quad O.RW \quad J.RW \rangle \quad \text{(C.19)}$$

For instance,



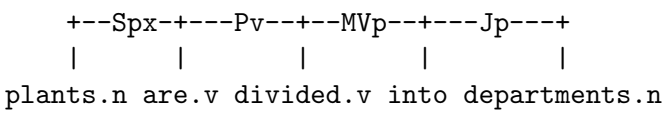
$$SF[sp]^* + O[spt]^* + M[v] + MV[sp]^* + J[sp]^* \;\Rightarrow$$
$$\langle M.RW\text{-}MV.RW \quad O.RW \quad J.RW \rangle \quad \text{(C.20)}$$

For instance,



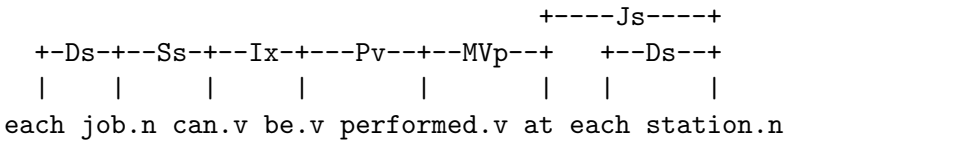
$$S[spx]^* + Pv + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle Pv.RW\text{-}MV.RW \quad S.LW \quad J.RW \rangle \quad \text{(C.21)}$$

For instance,



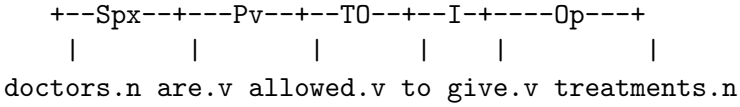
$$S[sp]^* + I[x]^* + Pv + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle Pv.RW\text{-}MV.RW \quad S.LW \quad J.RW \rangle \quad \text{(C.22)}$$

For instance,



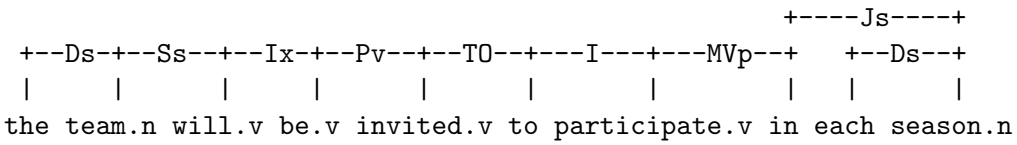
$$S[spx]^* + Pv + TO + I + O[sp]^* \;\Rightarrow\; \langle I.RW \quad S.LW \quad O.RW \rangle \quad \text{(C.23)}$$

For instance,



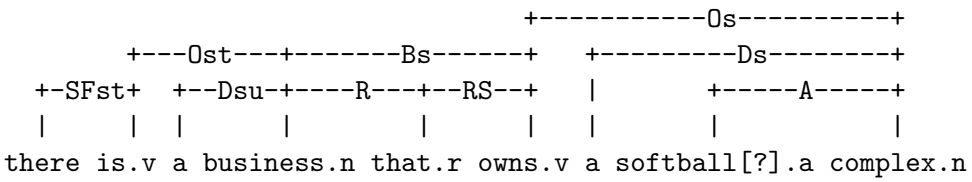
$$S[sp]^*+I[x]^*+Pv+TO+I+MV[sp]^*+J[sp]^* \;\Rightarrow\; \langle I.RW\text{-}MV.RW \quad S.LW \quad J.RW \rangle \quad \text{(C.24)}$$

For instance,



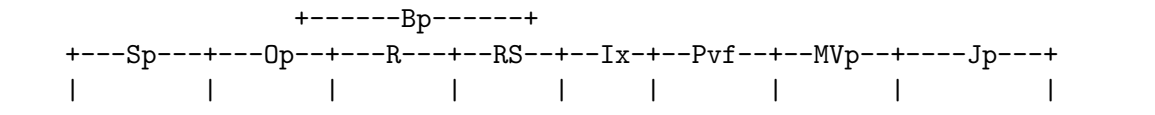
$$R + RS + O[sp]^* \;\Rightarrow\; \langle RS.RW \quad R.LW \quad O.RW \rangle \quad \text{(C.25)}$$

For instance,



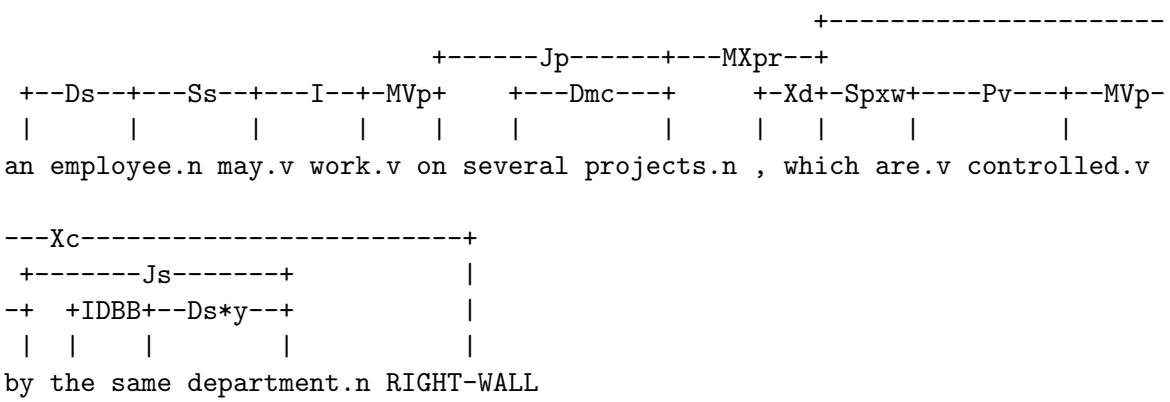
$$R+RS+I[x]^*+Pv[f]^*+MV[sp]^*+J[sp]^* \;\Rightarrow\; \langle Pv.RW\text{-}MV.RW \quad R.LW \quad J.RW \rangle \quad \text{(C.26)}$$

For instance,



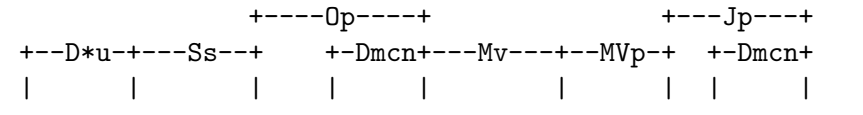
$$MX[spr]^* + S[spxw]^* + Pv + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle Pv.RW \quad MX.LW \quad J.RW \rangle \quad \text{(C.27)}$$

For instance,



$$Mv + MV[sp]^* + J[sp]^* \;\Rightarrow\; \langle Mv.RW\text{-}MV.RW \quad Mv.LW \quad J.RW \rangle \quad \text{(C.28)}$$

For instance,



APPENDIX D

TYPED DEPENDENCY-BASED HEURISTIC RULES

$$nsubj(w2, w1) + dobj(w2, w3) \;\Rightarrow\; \langle w2 \quad w1 \quad w3 \rangle \tag{D.1}$$

For instance,

```
each company has many plants.
```

```
det(company-2, each-1)                nsubj(has-3, company-2)
amod(plants-5, many-4)                dobj(has-3, plants-5)
```

$$nsubj(w2, w1) + prep(w2, w3) + pobj(w3, w4) \;\Rightarrow\; \langle w2\text{-}w3 \quad w1 \quad w4 \rangle \tag{D.2}$$

For instance,

```
each person works on some projects.
```

```
det(person-2, each-1)                 nsubj(works-3, person-2)
prep(works-3, on-4)                   det(projects-6, some-5)
pobj(on-4, projects-6)
```

$$nsubj(w2, w1) + xcomp(w2, w3) + dobj(w3, w4) \;\Rightarrow\; \langle w3 \quad w1 \quad w4 \rangle \tag{D.3}$$

For instance,

```
each student has to take at least one course.
```

```
det(student-2, each-1)                nsubj(has-3, student-2)
aux(take-5, to-4)                     xcomp(has-3, take-5)
dep(one-8, at-6)                      dep(one-8, least-7)
num(course-9, one-8)                  dobj(take-5, course-9)
```

$$nsubj(w2, w1)+xcomp(w2, w3)+prep(w3, w4)+pobj(w4, w5) \Rightarrow \langle w3\text{-}w4 \quad w1 \quad w5\rangle \quad \text{(D.4)}$$

For instance,

```
advanced courses need to be taught by professors.
```

```
amod(courses-2, advanced-1)          nsubj(need-3, courses-2)
aux(taught-6, to-4)                  auxpass(taught-6, be-5)
xcomp(need-3, taught-6)              prep(taught-6, by-7)
pobj(by-7, professors-8)
```

$$nsubj(w4, w1) + aux(w4, w2) + cop(w4, w3) \Rightarrow \langle w3 \quad w1 \quad w4\rangle \quad \text{(D.5)}$$

For instance,

```
An employee may also be a player.
```

```
det(employee-2, An-1)                nsubj(player-7, employee-2)
aux(player-7, may-3)                 advmod(player-7, also-4)
cop(player-7, be-5)                  det(player-7, a-6)
```

$$expl(w2, w1) + nsubj(w2, w3) + prep(w3, w4) + pobj(w4, w5) \Rightarrow \langle w2\text{-}w4 \quad w3 \quad w5\rangle \quad \text{(D.6)}$$

For instance,

```
there are 100 departments in the company.
```

```
expl(are-2, there-1)                 num(departments-4, 100-3)
nsubj(are-2, departments-4)          prep(departments-4, in-5)
det(company-7, the-6)                pobj(in-5, company-7)
```

$$prep(w4, w1) + pobj(w1, w2) + expl(w4, w3) + nsubj(w4, w5) \Rightarrow \langle w4\text{-}w1 \quad w5 \quad w2\rangle \quad \text{(D.7)}$$

For instance,

```
in the company, there are 100 departments.
```

```
prep(are-6, in-1)                    det(company-3, the-2)
pobj(in-1, company-3)                expl(are-6, there-5)
num(departments-8, 100-7)            nsubj(are-6, departments-8)
```

$$expl(w3, w1) + cop(w3, w2) + prep(w3, w4) + pobj(w4, w5) \;\Rightarrow\; \langle w2\text{-}w4 \quad w3 \quad w5 \rangle \quad \text{(D.8)}$$

For instance,

```
there can be multiple branches in the same city.
```

```
expl(branches-5, there-1)              aux(branches-5, can-2)
cop(branches-5, be-3)                  amod(branches-5, multiple-4)
prep(branches-5, in-6)                 det(city-9, the-7)
amod(city-9, same-8)                   pobj(in-6, city-9)
```

$$prep(w5, w1) + pobj(w1, w2) + expl(w5, w3) + cop(w5, w4) \;\Rightarrow\; \langle w4\text{-}w2 \quad w5 \quad w2 \rangle \quad \text{(D.9)}$$

For instance,

```
in the same city, there can be multiple branches.
```

```
prep(branches-10, in-1)                det(city-4, the-2)
amod(city-4, same-3)                   pobj(in-1, city-4)
expl(branches-10, there-6)             aux(branches-10, can-7)
cop(branches-10, be-8)                 amod(branches-10, multiple-9)
```

$$expl(w2, w1) + nsubj(w2, w3) + partmod(w3, w4) + prep(w4, w5) + pobj(w5, w6) \;\Rightarrow\; \langle w4\text{-}w5 \quad w3 \quad w6 \rangle \quad \text{(D.10)}$$

For instance,

```
there are five worker unions represented in the company.
```

```
expl(are-2, there-1)                   num(unions-5, five-3)
nn(unions-5, worker-4)                 nsubj(are-2, unions-5)
partmod(unions-5, represented-6)       prep(represented-6, in-7)
det(company-9, the-8)                  pobj(in-7, company-9)
```

$$nsubjpass(w2, w1) + prep(w2, w3) + pobj(w3, w4) \;\Rightarrow\; \langle w2\text{-}w3 \quad w1 \quad w4 \rangle \tag{D.11}$$

For instance,

```
plants are divided into departments.
```

```
nsubjpass(divided-3, plants-1)          auxpass(divided-3, are-2)
prep(divided-3, into-4)                 pobj(into-4, departments-5)
```

$$nsubjpass(w2, w1) + agent(w2, w3) \;\Rightarrow\; \langle w2 \quad w3 \quad w1 \rangle \tag{D.12}$$

For instance,

```
each project is controlled by a department.
```

```
det(project-2, each-1)                  nsubjpass(controlled-4, project-2)
auxpass(controlled-4, is-3)             det(department-7, a-6)
agent(controlled-4, department-7
```

$$nsubjpass(w2, w1) + xcomp(w2, w3) + dobj(w3, w4) \;\Rightarrow\; \langle w3 \quad w1 \quad w4 \rangle \tag{D.13}$$

For instance,

```
doctors are allowed to give treatments.
```

```
nsubjpass(allowed-3, doctors-1)         auxpass(allowed-3, are-2)
aux(give-5, to-4)                       xcomp(allowed-3, give-5)
dobj(give-5, treatments-6)
```

$$nsubjpass(w2, w1) + purpcl(w2, w3) + prep(w3,w4) + pobj(w4, w5) \;\Rightarrow\; \langle w3\text{-}w4 \quad w1 \quad w5 \rangle \tag{D.14}$$

For instance,

```
the team will be invited to participate in each season.
```

```
det(team-2, the-1)                      nsubjpass(invited-5, team-2)
aux(invited-5, will-3)                  auxpass(invited-5, be-4)
aux(participate-7, to-6)                purpcl(invited-5, participate-7)
prep(participate-7, in-8)               det(season-10, each-9)
pobj(in-8, season-10)
```

$$nsubjpass(w2, w1) + xcomp(w2, w3) + prep(w3,w4) + pobj(w4, w5)$$
$$\Rightarrow \langle w3\text{-}w4 \quad w1 \quad w5 \rangle \tag{D.15}$$

For instance,

```
employees are allowed to work on serveral projects.
```

```
nsubjpass(allowed-3, employees-1)      auxpass(allowed-3, are-2)
aux(work-5, to-4)                      xcomp(allowed-3, work-5)
prep(work-5, on-6)                     amod(projects-8, serveral-7)
pobj(on-6, projects-8)
```

$$rcmod(w1, w2) + nsubj(w2, w3) + dobj(w2, w4) \Rightarrow \langle w2 \quad w1 \quad w4 \rangle \tag{D.16}$$

For instance,

```
there is a business that owns a softball complex.
```

```
expl(is-2, there-1)                    det(business-4, a-3)
nsubj(is-2, business-4)                nsubj(owns-6, that-5)
rcmod(business-4, owns-6)              det(complex-9, a-7)
amod(complex-9, softball-8)            dobj(owns-6, complex-9)
```

$$rcmod(w1, w3) + nsubjpass(w3, w2) + prep(w3,w4) + pobj(w4, w5)$$
$$\Rightarrow \langle w3\text{-}w4 \quad w1 \quad w5 \rangle \tag{D.17}$$

For instance,

```
an employee may work on several projects, which are controlled by the
same department.
```

```
det(employee-2, an-1)                  nsubj(work-4, employee-2)
aux(work-4, may-3)                     prep(work-4, on-5)
amod(projects-7, several-6)            pobj(on-5, projects-7)
nsubjpass(controlled-11, which-9)      auxpass(controlled-11, are-10)
rcmod(projects-7, controlled-11)       prep(controlled-11, by-12)
det(department-15, the-13)             amod(department-15, same-14)
pobj(by-12, department-15)
```

$$partmod(w1, w2) + prep(w2, w3) + pobj(w3, w4) \;\Rightarrow\; \langle w2\text{-}w3 \quad w1 \quad w4 \rangle \tag{D.18}$$

For instance,

```
the company has 50 plants located in 40 states.
```

```
det(company-2, the-1)                  nsubj(has-3, company-2)
num(plants-5, 50-4)                    dobj(has-3, plants-5)
partmod(plants-5, located-6)           prep(located-6, in-7)
num(states-9, 40-8)                    pobj(in-7, states-9)
```

$$partmod(w1, w2) + dobj(w2, w3) \;\Rightarrow\; \langle w2 \quad w1 \quad w3 \rangle \tag{D.19}$$

For instance,

```
Each season consists of several leagues and teams, with each team
playing several games.
```

```
det(season-2, Each-1)                  nsubj(consists-3, season-2)
prep(consists-3, of-4)                 amod(leagues-6, several-5)
pobj(of-4, leagues-6)                  cc(leagues-6, and-7)
conj(leagues-6, teams-8)               prep(consists-3, with-10)
det(team-12, each-11)                  pobj(with-10, team-12)
partmod(team-12, playing-13)           amod(games-15, several-14)
dobj(playing-13, games-15)
```

APPENDIX E

AN ENTITY ATTACHMENT EXAMPLE

The following are the top three parses from the Link Parser for the sentence "the company has 50 plants located in 40 states and approximately 100,000 employees."

```
 Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=1 LEN=19)
 Sublinkage 1 (because of the coordinating conjunction):

                +----Op----+               +---Jp---+
 +--D*u-+---Ss--+    +-Dmcn+---Mv---+--MVp-+  +-Dmcn+
 |      |       |    |     |        |      |  |     |
the company.n has.v 50 plants.n located.v in 40 states.n and approximately


100,000 employees.n


Sublinkage 2 (because of the coordinating conjunction):

                +----Op----+               +---------------------Jp--------
 +--D*u-+---Ss--+    +-Dmcn+---Mv---+--MVp-+                       +----EN-
 |      |       |    |     |        |      |                       |
the company.n has.v 50 plants.n located.v in 40 states.n and approximately


-------------+
---+---Dmcn--+
   |         |
100,000 employees.n
```

```
Linkage 2, cost vector = (UNUSED=0 DIS=0 AND=1 LEN=22)
Sublinkage 1 (because of the coordinating conjunction):
                +------------MVp-----------+
                +----Op----+               +---Jp---+
 +--D*u-+---Ss--+    +-Dmcn+---Mv---+      |  +-Dmcn+
 |      |       |    |     |        |      |  |     |
the company.n has.v 50 plants.n located.v in 40 states.n and approximately


100,000 employees.n

Sublinkage 2 (because of the coordinating conjunction):

                +------------MVp-----------+
                +----Op----+               +----------------------Jp--------
 +--D*u-+---Ss--+    +-Dmcn+---Mv---+      |                        +----EN-
 |      |       |    |     |        |      |                        |
the company.n has.v 50 plants.n located.v in 40 states.n and approximately


-------------+
---+---Dmcn--+
   |         |
100,000 employees.n


Linkage 3, cost vector = (UNUSED=0 DIS=0 AND=3 LEN=27)
Sublinkage 1 (because of the coordinating conjunction):
                +----Op----+               +---Jp---+
 +--D*u-+---Ss--+    +-Dmcn+---Mv---+--MVp-+  +-Dmcn+
 |      |       |    |     |        |      |  |     |
the company.n has.v 50 plants.n located.v in 40 states.n and approximately


100,000 employees.n

Sublinkage 2 (because of the coordinating conjunction):
                +-----------------------------------Op---------------------
 +--D*u-+---Ss--+                                                    +----EN-
 |      |       |                                                    |
the company.n has.v 50 plants.n located.v in 40 states.n and approximately


-------------+
---+---Dmcn--+
   |         |
100,000 employees.n
```

APPENDIX F

CLIPS RULES

```
(defrule nsubj_dobj
  "extract subject-direct object tuple"
  (td (td_name nsubj) (word1 ?v1) (position1 ?p1) (word2 ?n1)
                      (position2 ?p0) (sent_id ?s1))
  (td (td_name dobj)  (word1 ?v1) (position1 ?p1) (word2 ?n2)
                      (position2 ?p2) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (entity1 ?n1) (entity1_position ?p0)
                 (entity2 ?n2) (entity2_position ?p2)
                 (rule a0) (sid ?s1))))


(defrule nsubj_prep
  "extract subject-direct object tuple"
  (td (td_name nsubj) (word1 ?v1) (position1 ?p1) (word2 ?n1)
                      (position2 ?p0) (sent_id ?s1))
  (td (td_name prep)  (word1 ?v1) (position1 ?p1) (word2 ?w1)
                      (position2 ?p2) (sent_id ?s1))
  (td (td_name pobj)  (word1 ?w1) (position1 ?p2) (word2 ?n2)
                      (position2 ?p3) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (relation_supplement ?w1)
                 (relation_supplement_position ?p2)
                 (entity1 ?n1) (entity1_position ?p0)
                 (entity2 ?n2) (entity2_position ?p3)
                 (rule a1) (sid ?s1))))
```

```
(defrule nsubj_xcomp_dobj
  "extract subject xcomp direct object tuple"
  (td (td_name nsubj) (word1 ?v1) (position1 ?p1) (word2 ?n1)
                      (position2 ?p0) (sent_id ?s1))
  (td (td_name xcomp) (word1 ?v1) (position1 ?p1) (word2 ?v2)
                      (position2 ?p2) (sent_id ?s1))
  (td (td_name dobj)  (word1 ?v2) (position1 ?p2) (word2 ?n3)
                      (position2 ?p3) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3))
  =>
  (assert (tuple (relation ?v2) (relation_position ?p2)
                 (entity1 ?n1) (entity1_position ?p0)
                 (entity2 ?n3) (entity2_position ?p3)
                 (rule a2) (sid ?s1))))


(defrule nsubj_xcomp_prep_pobj
  "extract subject xcomp prep pobj tuple"
  (td (td_name nsubj) (word1 ?v1) (position1 ?p1) (word2 ?n1)
                      (position2 ?p0) (sent_id ?s1))
  (td (td_name xcomp) (word1 ?v1) (position1 ?p1) (word2 ?v2)
                      (position2 ?p2) (sent_id ?s1))
  (td (td_name prep)  (word1 ?v2) (position1 ?p2) (word2 ?w1)
                      (position2 ?p3) (sent_id ?s1))
  (td (td_name pobj)  (word1 ?w1) (position1 ?p3) (word2 ?n2)
                      (position2 ?p4) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3 ?p4))
  =>
  (assert (tuple (relation ?v2) (relation_position ?p2)
                 (relation_supplement ?w1)
                 (relation_supplement_position ?p3)
                 (entity1 ?n1) (entity1_position ?p0)
                 (entity2 ?n2) (entity2_position ?p4)
                 (rule a3) (sid ?s1))))



(defrule nsubj_aux_cop
  "extract subject aux cop hierarchical structure, IS-A"
  (td (td_name nsubj) (word1 ?n2) (position1 ?p4) (word2 ?n1)
                      (position2 ?p1) (sent_id ?s1))
  (td (td_name aux)   (word1 ?n2) (position1 ?p4) (word2 ?v1)
                      (position2 ?p2) (sent_id ?s1))
  (td (td_name cop)   (word1 ?n2) (position1 ?p4) (word2 ?v2)
                      (position2 ?p3) (sent_id ?s1))
   (test (< ?p1 ?p2 ?p3 ?p4))
  =>
```

```
  (assert (tuple (relation ?v2) (relation_position ?p3)
                 (entity1 ?n1) (entity1_position ?p1)
                 (entity2 ?n2) (entity2_position ?p4)
                 (rule a4) (sid ?s1))))

;;-------------------------------------------------------------
;; Expletive sentence rules

(defrule expl_nsubj_prep_pobj
  "extract expletive tuple"
  (td (td_name expl)  (word1 ?v1) (position1 ?p1) (word2 ?w1)
                      (position2 ?p0) (sent_id ?s1))
  (td (td_name nsubj) (word1 ?v1) (position1 ?p1) (word2 ?n1)
                      (position2 ?p2) (sent_id ?s1))
  (td (td_name prep)  (word1 ?n1) (position1 ?p2) (word2 ?w2)
                      (position2 ?p3) (sent_id ?s1))
  (td (td_name pobj)  (word1 ?w2) (position1 ?p3) (word2 ?n2)
                      (position2 ?p4) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3 ?p4))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (relation_supplement ?w2)
                 (relation_supplement_position ?p3)
                 (entity1 ?n1) (entity1_position ?p2)
                 (entity2 ?n2) (entity2_position ?p4)
                 (rule e0) (sid ?s1))))


(defrule prep_pobj_expl_nsubj
  "extract expletive tuple"
  (td (td_name prep)  (word1 ?v1) (position1 ?p3) (word2 ?w1)
                      (position2 ?p0) (sent_id ?s1))
  (td (td_name pobj)  (word1 ?w1) (position1 ?p0) (word2 ?n1)
                      (position2 ?p1) (sent_id ?s1))
  (td (td_name expl)  (word1 ?v1) (position1 ?p3) (word2 ?w2)
                      (position2 ?p2) (sent_id ?s1))
  (td (td_name nsubj) (word1 ?v1) (position1 ?p3) (word2 ?n2)
                      (position2 ?p4) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3 ?p4))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p3)
                 (relation_supplement ?w1)
                 (relation_supplement_position ?p0)
                 (entity1 ?n2) (entity1_position ?p4)
                 (entity2 ?n1) (entity2_position ?p1)
                 (rule e1) (sid ?s1))))
```

```
(defrule expl_cop_prep_pobj
  "extract expletive tuple"
  (td (td_name expl)  (word1 ?n1) (position1 ?p2) (word2 ?w1)
                      (position2 ?p0) (sent_id ?s1))
  (td (td_name cop)   (word1 ?n1) (position1 ?p2) (word2 ?v1)
                      (position2 ?p1) (sent_id ?s1))
  (td (td_name prep)  (word1 ?n1) (position1 ?p2) (word2 ?w2)
                      (position2 ?p3) (sent_id ?s1))
  (td (td_name pobj)  (word1 ?w2) (position1 ?p3) (word2 ?n2)
                      (position2 ?p4) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3 ?p4))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (relation_supplement ?w2)
                 (relation_supplement_position ?p3)
                 (entity1 ?n1) (entity1_position ?p2)
                 (entity2 ?n2) (entity2_position ?p4)
                 (rule e2) (sid ?s1))))


(defrule prep_pobj_exp1_cop
  "extract expletive tuple"
  (td (td_name prep)  (word1 ?n1) (position1 ?p4) (word2 ?w1)
                      (position2 ?p0) (sent_id ?s1))
  (td (td_name pobj)  (word1 ?w1) (position1 ?p0) (word2 ?n2)
                      (position2 ?p1) (sent_id ?s1))
  (td (td_name expl)  (word1 ?n1) (position1 ?p4) (word2 ?w2)
                      (position2 ?p2) (sent_id ?s1))
  (td (td_name cop)   (word1 ?n1) (position1 ?p4) (word2 ?v1)
                      (position2 ?p3) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3 ?p4))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p3)
                 (relation_supplement ?w1)
                 (relation_supplement_position ?p0)
                 (entity1 ?n1) (entity1_position ?p4)
                 (entity2 ?n2) (entity2_position ?p1)
                 (rule e3) (sid ?s1))))


(defrule expl_nsubj_partmod_prep_pobj
  "extract expletive tuple"
  (td (td_name expl)      (word1 ?v1) (position1 ?p1) (word2 ?w1)
                          (position2 ?p0) (sent_id ?s1))
  (td (td_name nsubj)     (word1 ?v1) (position1 ?p1) (word2 ?n1)
                          (position2 ?p2) (sent_id ?s1))
```

```
  (td (td_name partmod)   (word1 ?n1) (position1 ?p2) (word2 ?v2)
                          (position2 ?p3) (sent_id ?s1))
  (td (td_name prep)      (word1 ?v2) (position1 ?p3) (word2 ?w2)
                          (position2 ?p4) (sent_id ?s1))
  (td (td_name pobj)      (word1 ?w2) (position1 ?p4) (word2 ?n2)
                          (position2 ?p5) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3 ?p4 ?p5))
  =>
  (assert (tuple (relation ?v2) (relation_position ?p3)
                 (relation_supplement ?w2)
                 (relation_supplement_position ?p4)
                 (entity1 ?n1) (entity1_position ?p2)
                 (entity2 ?n2) (entity2_position ?p5)
                 (rule e4) (sid ?s1))))


;; expletive with coordinating conjuction (specific)
(defrule expl_nsubj_partmod_prep_pobj
  "extract expletive tuple there be x and y in z"
  (td (td_name expl)      (word1 ?v1) (position1 ?p1) (word2 ?w1)
                          (position2 ?p0) (sent_id ?s1))
  (td (td_name nsubj)     (word1 ?v1) (position1 ?p1) (word2 ?n1)
                          (position2 ?p2) (sent_id ?s1))
  (td (td_name cc)        (word1 ?n1) (position1 ?p2) (word2 ?w0)
                          (position2 ?p3) (sent_id ?s1))
  (td (td_name conj)      (word1 ?n1) (position1 ?p2) (word2 ?n2)
                          (position2 ?p4) (sent_id ?s1))
  (td (td_name prep)      (word1 ?n2) (position1 ?p4) (word2 ?w2)
                          (position2 ?p5) (sent_id ?s1))
  (td (td_name pobj)      (word1 ?w2) (position1 ?p5) (word2 ?n3)
                          (position2 ?p6) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3 ?p4 ?p5 ?p6))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (relation_supplement ?w2)
                 (relation_supplement_position ?p5)
                 (entity1 ?n1) (entity1_position ?p2)
                 (entity2 ?n3) (entity2_position ?p6)
                 (rule e5) (sid ?s1))))
```

```
;;-------------------------------------------------------------
;; Passive sentence rules

(defrule nsubjpass_prep_pobj
  "extract passive subj prep pobj tuple"
  (td (td_name nsubjpass) (word1 ?v1) (position1 ?p1)
                          (word2 ?n1) (position2 ?p0) (sent_id ?s1))
  (td (td_name prep)      (word1 ?v1) (position1 ?p1) (word2 ?w1)
                          (position2 ?p2) (sent_id ?s1))
  (td (td_name pobj)      (word1 ?w1) (position1 ?p2) (word2 ?n2)
                          (position2 ?p3) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (relation_supplement ?w1)
                 (relation_supplement_position ?p2)
                 (entity1 ?n1) (entity1_position ?p0)
                 (entity2 ?n2) (entity2_position ?p3)
                 (rule p0) (sid ?s1))))

(defrule nsubjpass_agent
  "extract passive subjpass agent tuple"
  (td (td_name nsubjpass) (word1 ?v1) (position1 ?p1) (word2 ?n1)
                          (position2 ?p0) (sent_id ?s1))
  (td (td_name agent)     (word1 ?v1) (position1 ?p1) (word2 ?n2)
                          (position2 ?p2) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (entity1 ?n2) (entity1_position ?p2)
                 (entity2 ?n1) (entity2_position ?p0)
                 (rule p1) (sid ?s1))))

(defrule nsubjpass_xcomp_dobj
  "extract passive subjpass xcomp dobj tuple"
  (td (td_name nsubjpass) (word1 ?v1) (position1 ?p1) (word2 ?n1)
                          (position2 ?p0) (sent_id ?s1))
  (td (td_name xcomp)     (word1 ?v1) (position1 ?p1) (word2 ?v2)
                          (position2 ?p2) (sent_id ?s1))
  (td (td_name dobj)      (word1 ?v2) (position1 ?p2) (word2 ?n2)
                          (position2 ?p3) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3))
  =>
  (assert (tuple (relation ?v2) (relation_position ?p2)
                 (entity1 ?n1) (entity1_position ?p0)
                 (entity2 ?n2) (entity2_position ?p3)
                 (rule p2) (sid ?s1))))
```

```
(defrule nsubjpass_purpcl_prep_pobj
  "extract passive subjpass purpcl tuple"
  (td (td_name nsubjpass) (word1 ?v1) (position1 ?p1) (word2 ?n1)
                          (position2 ?p0) (sent_id ?s1))
  (td (td_name purpcl)    (word1 ?v1) (position1 ?p1) (word2 ?v2)
                          (position2 ?p2) (sent_id ?s1))
  (td (td_name prep)      (word1 ?v2) (position1 ?p2) (word2 ?w1)
                          (position2 ?p3) (sent_id ?s1))
  (td (td_name pobj)      (word1 ?w1) (position1 ?p3) (word2 ?n2)
                          (position2 ?p4) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3 ?p4))
  =>
  (assert (tuple (relation ?v2) (relation_position ?p2)
                 (relation_supplement ?w1)
                 (relation_supplement_position ?p3)
                 (entity1 ?n1) (entity1_position ?p0)
                 (entity2 ?n2) (entity2_position ?p4)
                 (rule p3) (sid ?s1))))


(defrule nsubjpass_xcomp_prep_pobj
  "extract passive subjpass xcomp pobj tuple"
  (td (td_name nsubjpass) (word1 ?v1) (position1 ?p1) (word2 ?n1)
                          (position2 ?p0) (sent_id ?s1))
  (td (td_name xcomp)     (word1 ?v1) (position1 ?p1) (word2 ?v2)
                          (position2 ?p2) (sent_id ?s1))
  (td (td_name prep)      (word1 ?v2) (position1 ?p2) (word2 ?w1)
                          (position2 ?p3) (sent_id ?s1))
  (td (td_name pobj)      (word1 ?w1) (position1 ?p3) (word2 ?n2)
                          (position2 ?p4) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3 ?p4))
  =>
  (assert (tuple (relation ?v2) (relation_position ?p2)
                 (relation_supplement ?w1)
                 (relation_supplement_position ?p3)
                 (entity1 ?n1) (entity1_position ?p0)
                 (entity2 ?n2) (entity2_position ?p4)
                 (rule p4) (sid ?s1))))
```

```
;;-------------------------------------------------------------
;; Subordinate clause and other rules

(defrule rcmod_nsubj_dobj
  "relative clasue tuple"
  (td (td_name rcmod)     (word1 ?n0) (position1 ?p0) (word2 ?v1)
                          (position2 ?p2) (sent_id ?s1))
  (td (td_name nsubj)     (word1 ?v1) (position1 ?p2)
                          (word2 ?n1&:(member$ ?n1 ?*dclause*))
                          (position2 ?p1) (sent_id ?s1))
  (td (td_name dobj)      (word1 ?v1) (position1 ?p2) (word2 ?n2)
                          (position2 ?p3) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p2)
                 (entity1 ?n0) (entity1_position ?p0)
                 (entity2 ?n2) (entity2_position ?p3)
                 (rule s0) (sid ?s1))))


(defrule rcmod_nsubjpass_prep_pobj
  "relative clasue tuple"
  (td (td_name rcmod)     (word1 ?n0) (position1 ?p0) (word2 ?v1)
                          (position2 ?p2) (sent_id ?s1))
  (td (td_name nsubjpass) (word1 ?v1) (position1 ?p2)
                          (word2 ?n1&:(member$ ?n1 ?*dclause*))
                          (position2 ?p1) (sent_id ?s1))
  (td (td_name prep)      (word1 ?v1) (position1 ?p2) (word2 ?w1)
                          (position2 ?p3) (sent_id ?s1))
  (td (td_name pobj)      (word1 ?w1) (position1 ?p3) (word2 ?n2)
                          (position2 ?p4) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3 ?p4))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p2)
                 (relation_supplement ?w1)
                 (relation_supplement_position ?p3)
                 (entity1 ?n0) (entity1_position ?p0)
                 (entity2 ?n2) (entity2_position ?p4)
                 (rule s1) (sid ?s1))))


(defrule partmod_prep_pobj
  "past particle modification"
  (td (td_name partmod)   (word1 ?n0) (position1 ?p0) (word2 ?v1)
                          (position2 ?p1) (sent_id ?s1))
  (td (td_name prep)      (word1 ?v1) (position1 ?p1) (word2 ?w1)
                          (position2 ?p2) (sent_id ?s1))
```

```
  (td (td_name pobj)      (word1 ?w1) (position1 ?p2) (word2 ?n1)
                          (position2 ?p3) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2 ?p3))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (relation_supplement ?w1)
                 (relation_supplement_position ?p2)
                 (entity1 ?n0) (entity1_position ?p0)
                 (entity2 ?n1) (entity2_position ?p3)
                 (rule s2) (sid ?s1))))


(defrule partmod_dobj
  "verb + ing"
  (td (td_name partmod)   (word1 ?n0) (position1 ?p0) (word2 ?v1)
                          (position2 ?p1) (sent_id ?s1))
  (td (td_name dobj)      (word1 ?v1) (position1 ?p1) (word2 ?n1)
                          (position2 ?p2) (sent_id ?s1))
  (test (< ?p0 ?p1 ?p2))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (entity1 ?n0) (entity1_position ?p0)
                 (entity2 ?n1) (entity2_position ?p2)
                 (rule s3) (sid ?s1))))




;;-----------------------------------------------------------------
;; of structure

(defrule of_nsubj
  "of structure rule of subject"
  (td (td_name nsubj)     (word1 ?v1) (position1 ?p4) (word2 ?n1)
                          (position2 ?p1) (sent_id ?s1))
  (td (td_name prep)      (word1 ?n1) (position1 ?p1) (word2 "of")
                          (position2 ?p2) (sent_id ?s1))
  (td (td_name pobj)      (word1 "of") (position1 ?p2) (word2 ?n2)
                          (position2 ?p3) (sent_id ?s1))
  (tuple (relation ?v1) (relation_position ?p4)
         (relation_supplement ?rs)
         (relation_supplement_position ?rsp)
         (entity1 ?n1) (entity1_position ?p1)
         (entity2 ?e2) (entity2_position ?e2p)
         (rule ?ru) (sid ?s1))
  (test (< ?p1 ?p2 ?p3 ?p4))
  =>
```

```
  (assert (tuple (relation ?v1) (relation_position ?p4)
                 (relation_supplement ?rs)
                 (relation_supplement_position ?rsp)
                 (entity1 ?n2) (entity1_position ?p3)
                 (entity2 ?e2) (entity2_position ?e2p)
                 (rule (sym-cat "of1-" ?ru)) (sid ?s1))))


(defrule of_nsubjpass
  "of structure rule of subject passive"
  (td (td_name nsubjpass) (word1 ?v1) (position1 ?p4) (word2 ?n1)
                          (position2 ?p1) (sent_id ?s1))
  (td (td_name prep)      (word1 ?n1) (position1 ?p1) (word2 "of")
                          (position2 ?p2) (sent_id ?s1))
  (td (td_name pobj)      (word1 "of") (position1 ?p2) (word2 ?n2)
                          (position2 ?p3) (sent_id ?s1))
  (tuple (relation ?v1) (relation_position ?p4)
         (relation_supplement ?rs)
         (relation_supplement_position ?rsp)
         (entity1 ?n1) (entity1_position ?p1)
         (entity2 ?e2) (entity2_position ?e2p)
         (rule ?ru) (sid ?s1))
  (test (< ?p1 ?p2 ?p3 ?p4))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p4)
                 (relation_supplement ?rs)
                 (relation_supplement_position ?rsp)
                 (entity1 ?n2) (entity1_position ?p3)
                 (entity2 ?e2) (entity2_position ?e2p)
                 (rule (sym-cat "of2-" ?ru)) (sid ?s1))))

(defrule of_dobj
  "of structure rule of direct object"
  (td (td_name dobj) (word1 ?v1) (position1 ?p1) (word2 ?e2)
                     (position2 ?e2p) (sent_id ?s1))
  (td (td_name prep) (word1 ?e2) (position1 ?e2p) (word2 "of")
                     (position2 ?p2) (sent_id ?s1))
  (td (td_name pobj) (word1 "of") (position1 ?p2) (word2 ?n2)
                     (position2 ?p3) (sent_id ?s1))
  (tuple (relation ?v1) (relation_position ?p1)
         (relation_supplement ?rs)
         (relation_supplement_position ?rsp)
         (entity1 ?e1) (entity1_position ?e1p)
         (entity2 ?e2) (entity2_position ?e2p)
         (rule ?ru) (sid ?s1))
  (test (< ?p1 ?e2p ?p2 ?p3))
  =>
```

```
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (relation_supplement ?rs)
                 (relation_supplement_position ?rsp)
                 (entity1 ?e1) (entity1_position ?e1p)
                 (entity2 ?n2) (entity2_position ?p3)
                 (rule (sym-cat "of3-" ?ru)) (sid ?s1))))

(defrule of_pobj
  "of structure rule of prepostional object"
  (td (td_name prep) (word1 ?v1) (position1 ?p1) (word2 ?rs)
                     (position2 ?rsp) (sent_id ?s1))
  (td (td_name pobj) (word1 ?rs) (position1 ?rsp) (word2 ?e2)
                     (position2 ?e2p) (sent_id ?s1))
  (td (td_name prep) (word1 ?e2) (position1 ?e2p) (word2 "of")
                     (position2 ?p3) (sent_id ?s1))
  (td (td_name pobj) (word1 "of") (position1 ?p3) (word2 ?n2)
                     (position2 ?p4) (sent_id ?s1))
  (tuple (relation ?v1) (relation_position ?p1)
         (relation_supplement ?rs)
         (relation_supplement_position ?rsp)
         (entity1 ?e1) (entity1_position ?e1p)
         (entity2 ?e2) (entity2_position ?e2p)
         (rule ?ru) (sid ?s1))
  (test (< ?p1 ?rsp ?e2p ?p3 ?p4))
  =>
  (assert (tuple (relation ?v1) (relation_position ?p1)
                 (relation_supplement ?rs)
                 (relation_supplement_position ?rsp)
                 (entity1 ?e1) (entity1_position ?e1p)
                 (entity2 ?n2) (entity2_position ?p4)
                 (rule (sym-cat "of4-" ?ru)) (sid ?s1))))
;;-----------------------------------------------------------------
;;Coordinating conjunctions
(defrule cc_subject
  "coordinating conjunction on subject (entity1)"
  (td (td_name cc)   (word1 ?e1) (position1 ?e1p) (word2 ?w0)
                     (position2 ?p1) (sent_id ?s1))
  (td (td_name conj) (word1 ?e1) (position1 ?e1p) (word2 ?n2)
                     (position2 ?p2) (sent_id ?s1))
  (tuple (relation ?r) (relation_position ?rp)
         (relation_supplement ?rs)
         (relation_supplement_position ?rsp)
         (entity1 ?e1) (entity1_position ?e1p)
         (entity2 ?e2) (entity2_position ?e2p)
         (sid ?s1))
  (test (< ?e1p ?p1 ?p2))
  =>
```

```
  (assert (tuple (relation ?r) (relation_position ?rp)
                 (relation_supplement ?rs)
                 (relation_supplement_position ?rsp)
                 (entity1 ?n2) (entity1_position ?p2)
                 (entity2 ?e2) (entity2_position ?e2p)
                 (rule cc1) (sid ?s1))))


(defrule cc_object
  "coordinating conjunction on object (entity2)"
  (td (td_name cc)   (word1 ?e2) (position1 ?e2p) (word2 ?w0)
                     (position2 ?p1) (sent_id ?s1))
  (td (td_name conj) (word1 ?e2) (position1 ?e2p) (word2 ?n2)
                     (position2 ?p2) (sent_id ?s1))
  (tuple (relation ?r) (relation_position ?rp)
         (relation_supplement ?rs)
         (relation_supplement_position ?rsp)
         (entity1 ?e1) (entity1_position ?e1p)
         (entity2 ?e2) (entity2_position ?e2p)
         (sid ?s1))
  (test (< ?e1p ?e2p ?p1 ?p2))
  =>
  (assert (tuple (relation ?r) (relation_position ?rp)
                 (relation_supplement ?rs)
                 (relation_supplement_position ?rsp)
                 (entity1 ?e1) (entity1_position ?e1p)
                 (entity2 ?n2) (entity2_position ?p2)
                 (rule cc2) (sid ?s1))))

(defrule cc_verb_and_pobj
  "coordinating conjunction on verb (relation) and obj"
  (td (td_name nsubj)  (word1 ?r) (position1 ?rp) (word2 ?e1)
                       (position2 ?e1p) (sent_id ?s1))
  (td (td_name cc)     (word1 ?r) (position1 ?rp) (word2 ?w0)
                       (position2 ?p1) (sent_id ?s1))
  (td (td_name conj)   (word1 ?r) (position1 ?rp) (word2 ?v2)
                       (position2 ?p2) (sent_id ?s1))
  (td (td_name prep)   (word1 ?v2) (position1 ?p2) (word2 ?w1)
                       (position2 ?p3) (sent_id ?s1))
  (td (td_name pobj)   (word1 ?w1) (position1 ?p3) (word2 ?n2)
                       (position2 ?p4) (sent_id ?s1))
  (tuple (relation ?r) (relation_position ?rp)
         (entity1 ?e1) (entity1_position ?e1p)
         (entity2 ?e2) (entity2_position ?e2p)
         (sid ?s1))
  (test (< ?e1p ?rp ?e2p ?p1 ?p2 ?p3 ?p4))
  =>
```

```
  (assert (tuple (relation ?v2) (relation_position ?p2)
                 (relation_supplement ?w1)
                 (relation_supplement_position ?p3)
                 (entity1 ?e1) (entity1_position ?e1p)
                 (entity2 ?n2) (entity2_position ?p4)
                 (rule cc31) (sid ?s1))))

(defrule cc_verb_and_dobj
  "coordinating conjunction on verb (relation) and obj"
  (td (td_name nsubj) (word1 ?r) (position1 ?rp) (word2 ?e1)
                      (position2 ?e1p) (sent_id ?s1))
  (td (td_name cc)    (word1 ?r) (position1 ?rp) (word2 ?w0)
                      (position2 ?p1) (sent_id ?s1))
  (td (td_name conj)  (word1 ?r) (position1 ?rp) (word2 ?v2)
                      (position2 ?p2) (sent_id ?s1))
  (td (td_name dobj)  (word1 ?v2) (position1 ?p2) (word2 ?n2)
                      (position2 ?p3) (sent_id ?s1))
  (tuple (relation ?r) (relation_position ?rp)
         (entity1 ?e1) (entity1_position ?e1p)
         (entity2 ?e2) (entity2_position ?e2p)
         (sid ?s1))
  (test (< ?e1p ?rp ?e2p ?p1 ?p2 ?p3))
  =>
  (assert (tuple (relation ?v2) (relation_position ?p2)
                 (entity1 ?e1) (entity1_position ?e1p)
                 (entity2 ?n2) (entity2_position ?p3)
                 (rule cc32) (sid ?s1))))

;; this one preempt cc3
(defrule cc_verb_and_obj_further
  "coordinating conjunction on verb (relation) and obj further"
  (td (td_name cc)     (word1 ?r) (position1 ?rp) (word2 ?w0)
                       (position2 ?p1) (sent_id ?s1))
  (td (td_name advmod) (word1 ?v2) (position1 ?p3) (word2 "further")
                       (position2 ?p2) (sent_id ?s1))
  (td (td_name conj)   (word1 ?r) (position1 ?rp) (word2 ?v2)
                       (position2 ?p3) (sent_id ?s1))
  (td (td_name prep)   (word1 ?v2) (position1 ?p3) (word2 ?w1)
                       (position2 ?p4) (sent_id ?s1))
  (td (td_name pobj)   (word1 ?w1) (position1 ?p4) (word2 ?n2)
                       (position2 ?p5) (sent_id ?s1))
  (tuple (relation ?r) (relation_position ?rp)
                 (entity1 ?e1) (entity1_position ?e1p)
                 (entity2 ?e2) (entity2_position ?e2p)
                 (sid ?s1))
  (test (< ?e1p ?rp ?e2p ?p1 ?p2 ?p3 ?p4 ?p5))
  =>
```

```
  (assert (tuple (relation ?v2) (relation_position ?p3)
                                (relation_supplement ?w1)
                                (relation_supplement_position ?p4)
                                (entity1 ?e2) (entity1_position ?e2p)
                                (entity2 ?n2) (entity2_position ?p5)
                                (rule cc3f) (sid ?s1))) )


(defrule cc_preconj_dep
  "coordinating conjunction ... either ... or"
  (td (td_name preconj)   (word1 ?e2) (position1 ?e2p) (word2 ?w1)
                          (position2 ?p1) (sent_id ?s1))
  (td (td_name cc)        (word1 ?e2) (position1 ?e2p) (word2 ?w2)
                          (position2 ?p2) (sent_id ?s1))
  (td (td_name dep)       (word1 ?e2) (position1 ?e2p) (word2 ?n2)
                          (position2 ?p3) (sent_id ?s1))
  (tuple (relation ?r) (relation_position ?rp)
         (relation_supplement ?rs)
         (relation_supplement_position ?rsp)
         (entity1 ?e1) (entity1_position ?e1p)
         (entity2 ?e2) (entity2_position ?e2p)
         (sid ?s1))
  (test (< ?e1p ?rp ?p1 ?e2p ?p2 ?p3))
  =>
  (assert (tuple (relation ?r) (relation_position ?rp)
                 (relation_supplement ?rs)
                 (relation_supplement_position ?rsp)
                 (entity1 ?e1) (entity1_position ?e1p)
                 (entity2 ?n2) (entity2_position ?p3)
                 (rule cc4) (sid ?s1))))


;;-----------------------------------------------------------------
;;Compound noun rules

(defrule compound_noun_nn1
  "compound noun nn rule for entity1"
  ?t1 <- (tuple (entity1 ?e1) (entity1_position ?p1)
         (entity1_supplement "") (sid ?s1))
  (td (td_name nn) (word1 ?e1) (position1 ?p1) (word2 ?w0)
                   (position2 ?p0) (sent_id ?s1))
  =>
  (modify ?t1 (entity1_supplement ?w0) (entity1_supplement_position ?p0)))
```

```
(defrule compound_noun_nn2
  "compound noun nn rule for entity2"
  ?t1 <- (tuple (entity2 ?e1) (entity2_position ?p1)
         (entity2_supplement "") (sid ?s1))
  (td (td_name nn) (word1 ?e1) (position1 ?p1) (word2 ?w0)
                   (position2 ?p0) (sent_id ?s1))
  =>
  (modify ?t1 (entity2_supplement ?w0) (entity2_supplement_position ?p0)))


;;Cardinality according to POS
(defrule cardinality_one_NN1
  "determing entity1 cardinality according NN"
  ?t1 <- (tuple (entity1 ?e1) (entity1_position ?p)
         (entity1_cardinality nil) (sid ?s1))
  (unit (word ?e1) (position ?p) (pos NN) (sent_id ?s1))
  =>
  (modify ?t1 (entity1_cardinality ONE)))

(defrule cardinality_one_NN2
  "determing entity2 cardinality according NN"
  ?t1 <- (tuple (entity2 ?e2) (entity2_position ?p)
         (entity2_cardinality nil) (sid ?s1))
  (unit (word ?e2) (position ?p) (pos NN) (sent_id ?s1))
  =>
  (modify ?t1 (entity2_cardinality ONE)))

(defrule cardinality_many_NNS1
  "determing entity1 cardinality according NNS"
  ?t1 <- (tuple (entity1 ?e1) (entity1_position ?p)
         (entity1_cardinality nil) (sid ?s1))
  (unit (word ?e1) (position ?p) (pos NNS) (sent_id ?s1))
  =>
  (modify ?t1 (entity1_cardinality MANY)))

(defrule cardinality_one_NNS2
  "determing entity2 cardinality according NNS"
  ?t1 <- (tuple (entity2 ?e2) (entity2_position ?p)
         (entity2_cardinality nil) (sid ?s1))
  (unit (word ?e2) (position ?p) (pos NNS) (sent_id ?s1))
  =>
  (modify ?t1 (entity2_cardinality MANY)))
```

```
;; cardinality adjustments
;; "each student, every student"
(defrule cardinality_one_NN1_adjust1
  "adjust cardinality for exceptions"
  ?t1 <- (tuple (entity1_position ?p) (entity1_cardinality ONE) (sid ?s1))
  (unit (word "each") (position ?p1&:(eq ?p1 (- ?p 1))) (sent_id ?s1))
  =>
  (printout t "test" crlf)
  (modify ?t1 (entity1_cardinality MANY)))

(defrule cardinality_one_NN2_adjust1
  "adjust cardinality for exceptions"
  ?t1 <- (tuple (entity2_position ?p) (entity2_cardinality ONE) (sid ?s1))
  (unit (word "each") (position ?p1&:(= ?p1 (- ?p 1))) (sent_id ?s1))
  =>
  (modify ?t1 (entity2_cardinality MANY)))

(defrule cardinality_one_NN1_adjust2
  "adjust cardinality for exptions more_than_one, at_least_one"
  ?t1 <- (tuple (entity1_position ?p) (entity1_cardinality ONE) (sid ?s1))
  (unit (word ?w&:(member$ ?w ?*many*))
        (position ?p1&:(= ?p1 (- ?p 1))) (sent_id ?s1))
  =>
  (modify ?t1 (entity1_cardinality MANY)))

(defrule cardinality_one_NN2_adjust2
  "adjust cardinality for exceptions"
  ?t1 <- (tuple (entity2_position ?p) (entity2_cardinality ONE) (sid ?s1))
  (unit (word ?w&:(member$ ?w ?*many*))
        (position ?p1&:(= ?p1 (- ?p 1))) (sent_id ?s1))
  =>
  (modify ?t1 (entity2_cardinality MANY)))


;;-----------------------------------------------------------------------
;;get tuple base form
(defrule relation_base_form
  "get the base form of the relation"
  ?t1 <- (tuple (relation ?r) (relation_position ?rp)
         (sid ?s) (relation_base ""))
  (unit (word ?r) (position ?rp) (sent_id ?s) (root ?root))
  =>
  (modify ?t1 (relation_base ?root)))
```

```
(defrule entity1_base_form
  "get the base form of the entity1"
  ?t1 <- (tuple (entity1 ?e1) (entity1_position ?e1p)
         (sid ?s) (entity1_base ""))
  (unit  (word ?e1) (position ?e1p) (sent_id ?s) (root ?root))
  =>
  (modify ?t1 (entity1_base ?root)))

(defrule entity2_base_form
  "get the base form of the entity2"
  ?t1 <- (tuple (entity2 ?e2) (entity2_position ?e2p)
         (sid ?s) (entity2_base ""))
  (unit  (word ?e2) (position ?e2p) (sent_id ?s) (root ?root))
  =>
  (modify ?t1 (entity2_base ?root)))

;;-------------------------------------------------------------------
;; transform some of the entity to attributes
(defrule name_attribute1
  "if entity1 is in typical attributes, then transform the entity
   tuple to an attribute pair"
  ?t1 <- (tuple (entity1 ?e1) (entity1_position ?ep1)
                (entity1_base ?eb1) (entity1_supplement ?es1)
                (entity1_supplement_position ?esp1)
                (entity2 ?e2) (entity2_position ?ep2)
                (entity2_base ?eb2)
                (entity2_supplement ?es2)
                (entity2_supplement_position ?esp2))
  (test(member$ ?eb2 ?*typical_attribute*))
  =>
  (assert(entity_attribute (entity ?e1) (entity_position ?ep1)
                           (entity_base ?eb1) (entity_supplement ?es1)
                           (entity_supplement_position ?esp1)
                           (attribute ?e2) (attribute_position ?ep2)
                           (attribute_base ?eb2) (attribute_supplement ?es2)
                           (attribute_supplement_position ?esp2)))
  (retract ?t1))

(defrule name_attribute2
  "if entity2 is in typical attributes, then transform the entity
   tuple to an attribute pair"
  ?t1 <- (tuple (entity1 ?e1) (entity1_position ?ep1)
                (entity1_base ?eb1&:(member$ ?eb1 ?*typical_attribute*))
                (entity1_supplement ?es1)
                (entity1_supplement_position ?esp1)
                (entity2 ?e2) (entity2_position ?ep2)
                (entity2_base ?eb2)
```

```
               (entity2_supplement ?es2)
               (entity2_supplement_position ?esp2))
 =>
 (assert (entity_attribute (entity ?e2) (entity_position ?ep2)
                           (entity_base ?eb2) (entity_supplement ?es2)
                           (entity_supplement_position ?esp2)
                           (attribute ?e1) (attribute_position ?ep1)
                           (attribute_base ?eb1) (attribute_supplement ?es1)
                           (attribute_supplement_position ?esp1)))
 (retract ?t1))

(defrule verb_attribute_passive
  "if the relation is in typical attribute verbs, then transform the
   tuple to attribute"
  ?t1 <- (tuple (relation ?r) (relation_position ?rp)
                (relation_base ?rb&:(member$ ?rb ?*typical_attribute_verb*))
                (relation_supplement "by")
                (relation_supplement_position ?rsp)
                (entity1 ?e1) (entity1_position ?ep1)
                (entity1_base ?eb1) (entity1_supplement ?es1)
                (entity1_supplement_position ?esp1)
                (entity2 ?e2) (entity2_position ?ep2)
                (entity2_base ?eb2) (entity2_supplement ?es2)
                (entity2_supplement_position ?esp2))
 =>
  (assert (entity_attribute (entity ?e1) (entity_position ?ep1)
                            (entity_base ?eb1) (entity_supplement ?es1)
                            (entity_supplement_position ?esp1)
                            (attribute ?e2) (attribute_position ?ep2)
                            (attribute_base ?eb2) (attribute_supplement ?es2)
                            (attribute_supplement_position ?esp2)))
  (retract ?t1))

(defrule verb_attribute_active
  "if the relation is in typical attribute verbs, then transfrom the
   tuple to attribute"
  ?t1 <- (tuple (relation ?r) (relation_position ?rp)
                (relation_base ?rb&:(member$ ?rb ?*typical_attribute_verb*))
                (relation_supplement "")
                (relation_supplement_position ?rsp)
                (entity1 ?e1) (entity1_position ?ep1)
                (entity1_base ?eb1) (entity1_supplement ?es1)
                (entity1_supplement_position ?esp1)
                (entity2 ?e2) (entity2_position ?ep2)
                (entity2_base ?eb2) (entity2_supplement ?es2)
                (entity2_supplement_position ?esp2))
 =>
```

```
  (assert (entity_attribute (entity ?e2) (entity_position ?ep2)
                            (entity_base ?eb2) (entity_supplement ?es2)
                            (entity_supplement_position ?esp2)
                            (attribute ?e1) (attribute_position ?ep1)
                            (attribute_base ?eb1) (attribute_supplement ?es1)
                            (attribute_supplement_position ?esp1)))
  (retract ?t1))

(defrule database_attribute
  "if the entity is word database"
  ?t1 <- (entity_attribute (entity ?e) (entity_position ?ep)
                           (entity_base "database") (entity_supplement ?es)
                           (entity_supplement_position ?esp)
                           (attribute ?a) (attribute_position ?ap)
                           (attribute_base ?ab) (attribute_supplement ?as)
                           (attribute_supplement_position ?asp))
  =>
  (modify ?t1 (entity ?as) (entity_position ?asp) (entity_base ?as)
          (entity_supplement "") (entity_supplement_position -1)
          (attribute_supplement "") (attribute_supplement_position -1)))

;;-------------------------------------------------------------------------
;;combining tuples to triple
(defrule tuples_to_triple1
  "combining tuples to triples"
  (declare (salience -700))
  ?t1 <- (tuple (relation ?r) (relation_position ?rp)
                (relation_base ?rb)
                (relation_supplement "")
                (entity1 ?e1) (entity1_position ?ep1)
                (entity1_base ?eb1) (entity1_supplement ?es1)
                (entity1_supplement_position ?esp1)
                (entity1_cardinality ?ec1)
                (entity2 ?e2) (entity2_position ?ep2)
                (entity2_base ?eb2) (entity2_supplement ?es2)
                (entity2_supplement_position ?esp2)
                (entity2_cardinality ?ec2))
  ?t2 <- (tuple (relation ?r) (relation_position ?rp)
                (relation_base ?rb)
                (relation_supplement ?rs&:(neq ?rs ""))
                (relation_supplement_position ?rsp)
                (entity1 ?e1) (entity1_position ?ep1)
                (entity1_base ?eb1) (entity1_supplement ?es1)
                (entity1_supplement_position ?esp1)
                (entity1_cardinality ?ec1)
                (entity2 ?e3) (entity2_position ?ep3)
                (entity2_base ?eb3) (entity2_supplement ?es3)
```

```
                (entity2_supplement_position ?esp3)
                (entity2_cardinality ?ec3))
  =>
  (assert (triple (relation ?r) (relation_position ?rp)
                (relation_base ?rb)
                (relation_supplement ?rs)
                (relation_supplement_position ?rsp)
                (entity1 ?e1) (entity1_position ?ep1)
                (entity1_base ?eb1) (entity1_supplement ?es1)
                (entity1_supplement_position ?esp1)
                (entity1_cardinality ?ec1)
                (entity2 ?e2) (entity2_position ?ep2)
                (entity2_base ?eb2) (entity2_supplement ?es2)
                (entity2_supplement_position ?esp2)
                (entity2_cardinality ?ec2)
                (entity3 ?e3) (entity3_position ?ep3)
                (entity3_base ?eb3) (entity3_supplement ?es3)
                (entity3_supplement_position ?esp3)
                (entity3_cardinality ?ec3)))
(retract ?t1)
(retract ?t2))


(defrule tuples_to_triple12
  "combining tuples to triples"
  (declare (salience -700))
  ?t1 <- (tuple (relation ?r) (relation_position ?rp)
                (relation_base ?rb)
                (relation_supplement "")
                (entity1 ?e1) (entity1_position ?ep1)
                (entity1_base ?eb1) (entity1_supplement ?es1)
                (entity1_supplement_position ?esp1)
                (entity1_cardinality ?ec1)
                (entity2 ?e2) (entity2_position ?ep2)
                (entity2_base ?eb2) (entity2_supplement ?es2)
                (entity2_supplement_position ?esp2)
                (entity2_cardinality ?ec2))
  ?t2 <- (tuple (relation ?r) (relation_position ?rp)
                (relation_base ?rb)
                (relation_supplement ?rs&:(neq ?rs ""))
                (relation_supplement_position ?rsp)
                (entity1 ?e3) (entity1_position ?ep3)
                (entity1_base ?eb3) (entity1_supplement ?es3)
                (entity1_supplement_position ?esp3)
                (entity1_cardinality ?ec3)
                (entity2 ?e1) (entity2_position ?ep1)
                (entity2_base ?eb1) (entity2_supplement ?es1)
```

```
                (entity2_supplement_position ?esp1)
                (entity2_cardinality ?ec1))
  =>
(assert (triple (relation ?r) (relation_position ?rp)
                (relation_base ?rb)
                (relation_supplement ?rs)
                (relation_supplement_position ?rsp)
                (entity1 ?e1) (entity1_position ?ep1)
                (entity1_base ?eb1) (entity1_supplement ?es1)
                (entity1_supplement_position ?esp1)
                (entity1_cardinality ?ec1)
                (entity2 ?e2) (entity2_position ?ep2)
                (entity2_base ?eb2) (entity2_supplement ?es2)
                (entity2_supplement_position ?esp2)
                (entity2_cardinality ?ec2)
                (entity3 ?e3) (entity3_position ?ep3)
                (entity3_base ?eb3) (entity3_supplement ?es3)
                (entity3_supplement_position ?esp3)
                (entity3_cardinality ?ec3)))
(retract ?t1)
(retract ?t2))

;; filtering typical none entity tuples
(defrule of1_filtering
  "a tuple is replaced by of1 if entity1 is a none-entity "
  (declare (salience -800))
  ?t1 <- (tuple (relation ?r)
                (entity1 ?e1&:(member$ ?e1 ?*typical_none_entity*))
                (entity2 ?e2) (rule ?ru) (sid ?s))
         (tuple (relation ?r) (entity1 ?ex) (entity2 ?e2)
                (rule of1) (sid ?s))
  =>
  (retract ?t1))

(defrule of2_fitering
  "a tuple is replaced by of2 if entity1 is a none-entity "
  (declare (salience -800))
  ?t1 <- (tuple (relation ?r)
                (entity1 ?e1&:(member$ ?e1 ?*typical_none_entity*))
                (entity2 ?e2) (rule ?ru) (sid ?s))
         (tuple (relation ?r) (entity1 ?ex) (entity2 ?e2)
                (rule of2) (sid ?s))
  =>
  (retract ?t1))
```

```
(defrule of3_filtering
  "a tuple is replaced by of3 if entity2 is a none-entity "
  (declare (salience -800))
  ?t1 <- (tuple (relation ?r) (entity1 ?e1)
                (entity2 ?e2&:(member$ ?e2 ?*typical_none_entity*))
                (rule ?ru) (sid ?s))
         (tuple (relation ?r) (entity1 ?e1) (entity2 ?ex)
                (rule of3) (sid ?s))
  =>
  (retract ?t1))


(defrule of4_filtering
  "a tuple is replace by of4 if entity2 is a none-entity "
  (declare (salience -800))
  ?t1 <- (tuple (relation ?r) (entity1 ?e1)
                (entity2 ?e2&:(member$ ?e2 ?*typical_none_entity*))
                (rule ?ru) (sid ?s))
         (tuple (relation ?r) (entity1 ?e1) (entity2 ?ex)
                (rule of4) (sid ?s))
  =>
  (retract ?t1))


;;strong filters

(defrule typical_none_entity1_filter
  "a tuple is retracted if entity1 is a none-entity "
  (declare (salience -800))
  ?t1 <- (tuple (relation ?r)
                (entity1 ?e1&:(member$ ?e1 ?*typical_none_entity*))
                (entity2 ?e2) (rule ?ru) (sid ?s))
  =>
  (retract ?t1))


(defrule typical_none_entity2_filter
  "a tuple is retracted if  entity 2 is a none-entity "
  (declare (salience -800))
  ?t1 <- (tuple (relation ?r) (entity1 ?e1)
                (entity2 ?e2&:(member$ ?e2 ?*typical_none_entity*))
                (rule ?ru) (sid ?s))
  =>
  (retract ?t1))
```

```
;; specfic coordinating conjuction rules cc3f preempt cc3
(defrule cc3f_cc3
  "cc3f preempts cc2  "
  (declare (salience -800))
  ?t1 <- (tuple (relation ?r) (entity1 ?e1) (entity2 ?e2)
         (rule cc3) (sid ?s))
         (tuple (relation ?r) (entity1 ?ex) (entity2 ?e2)
         (rule cc3f) (sid ?s))
  =>
  (retract ?t1))



;; deduplication by base form

(defrule deduplication1
  "deduping same tuples "
  (declare (salience -800))
  ?t1 <- (tuple (relation_base ?r) (relation_supplement ?rs)
                (entity1_base ?e1) (entity1_supplement ?e1s)
                (entity2_base ?e2) (entity2_supplement ?e2s))
  ?t2 <- (tuple (relation_base ?r) (relation_supplement ?rs)
                (entity1_base ?e1) (entity1_supplement ?e1s)
                (entity2_base ?e2) (entity2_supplement ?e2s))
  (test (neq ?t1 ?t2))
  =>
  (retract ?t1))


(defrule deduplication2
  "deduping same tuples "
  (declare (salience -800))
  ?t1 <- (tuple (relation_base ?r) (relation_supplement ?rs)
                (entity1_base ?e1) (entity1_supplement ?e1s)
                (entity2_base ?e2) (entity2_supplement ?e2s))
  ?t2 <- (tuple (relation_base ?r) (relation_supplement ?rs)
                (entity1_base ?e2) (entity1_supplement ?e2s)
                (entity2_base ?e1) (entity2_supplement ?e1s))
  (test (neq ?t1 ?t2))
  =>
  (retract ?t1))
```

```
(defrule deduplication3
  "deduping same tuples "
  (declare (salience -800))
  ?t1 <- (tuple (relation_base ?r) (relation_supplement "")
                (entity1_base ?e1) (entity1_supplement ?e1s)
                (entity1_cardinality ?ec11)
                (entity2_base ?e2) (entity2_supplement ?e2s)
                (entity2_cardinality ?ec12)
                )
  ?t2 <- (tuple (relation_base ?r) (relation_supplement "by")
                (entity1_base ?e2) (entity1_supplement ?e2s)
                (entity1_cardinality ?ec21)
                (entity2_base ?e1) (entity2_supplement ?e1s)
                (entity2_cardinality ?ec22)
                )
  (test (neq ?t1 ?t2))
  =>
  (modify ?t1 (entity1_cardinality ?ec22))
  (retract ?t2))
```

APPENDIX G

EXPERIMENTAL DATABASE DESIGN PROBLEMS

G.1 EASIER

PROBLEM 1

A bookstore sells books and software to students. Each student has a student id, first name, last name, address, and phone number. A book has title, ISBN, author, publisher, number of copies in stock and price. Software has name, price, and required-platform. The bookstore hires employees. An employee has his employee id, name, address, and phone number. Students may also be employees.

PROBLEM 2

A library owns one or more copies of some books. Each customer can check out one or more books from libraries. A library has name, working hours, types, maximum limit of books that can be checked out and a address which contains street number, street name, city, zip code. A customer has a name, year of birth, ssn, phone number and address. A book has a name, year of publication, publisher, ISBN, price. Each check out transaction has a check out date, an expected return date and an actual return date.

PROBLEM 3

A bookstore sells books and software to students. Each student has a student id, first name, last name, address, and phone number. A book has title, ISBN, author, publisher, number

of copies in stock and price. Software has name, price, and required-platform. When a student buys books, he or she may get a discount. Discount type, purchase quantity and actual price should be stored in database. The bookstore hires employees. An employee has his employee id, name, address, and phone number. Students may also be employees.

PROBLEM 4

A concert season schedules one or more concerts. A concert season is identified by its opening date, which includes month, day, and year. A concert includes one or more compositions. A concert is identified by its number. A concert also has a concert date, which consists of month, day, year, and time. For each concert there is one conductor. A conductor is identified by his/her PID. A conductor name also needs to be included in the database.

PROBLEM 5

The company has 50 plants located in 40 states and approximately 100,000 employees. Each plant is divided into departments and further subdivided into work stations. There are 100 departments and 500 work stations in the company. A work task is associated with one of 20 different job types. There are five worker unions represented in the company, and every employee belongs to exactly one union.

PROBLEM 6

A database to maintain information about hospital staff and patients. Doctors and nurses are staff. Staff has name, address and social-security number. Patients have their names, addresses, and insurance company name. Patients are assigned to a ward (room). Nurses are assigned to zero or more wards. Each ward has at least one nurse assigned. Doctors are assigned to zero or more patients. Patients may or may not have a doctor assigned, and they may have more than one doctor. Patients in the same ward may have different doctors but will always have the same nurse(s).

PROBLEM 7

Students purchase books from a bookstore. Each student has a student id, first name, last name, address, and phone number. A book has title, ISBN, author, publisher, number of copies in stock and price. Each purchase transaction has a discount type, purchase quantity and actual price. The bookstore hires employees. An employee has his employee id, name, address, and phone number. Students may also be employees.

PROBLEM 8

Cell phone service plans are provided by different wireless service providers. A customer has a name and an address. Each customer can be an individual OR a business customer. Every individual has a year of birth, social security number and an age. Each business customer has a contact person name and a web address. A wireless service provider has a name and a web address. A service plan has a plan name and a start date. Any phone, associated with a service plan, is assigned to one customer. Any phone has an area code, a number, current balance.

PROBLEM 9

An online movie rental shop rents movies to customers. Each customer has a unique id, name, address, telephone number and a credit. Each movie has a unique id, and title, movie type, media type and duration. Every customer has an account in the system. Each account has a unique account id, account type and the balance on the account.

PROBLEM 10

Design a database for a bus company. Each bus route has a starting place, an ending place and several intermediate stops. The company is distributed over several branches. Each branch must be located along the bus routes. Each branch has a address, working hours and phone number. At least one bus is assigned to one bus route. Each bus has a plate number and a driver. The driver's name, address and phone number are also need to be kept in the database.

G.2 HARDER

PROBLEM 1

The company has 50 plants located in 40 states and approximately 100,000 employees. Each plant is divided into departments and further subdivided into work stations. There are 100 departments and 500 work stations in the company. A work task is associated with one of 20 different job types. Each of the job types can be performed at each of the plants. During a given day an employee may perform more than one work task, each associated with a different job type, and each can be performed at different work station. There are five worker unions represented in the company, and every employee belongs to exactly one union.

PROBLEM 2

Database for a software company. Each employee has name, date of birth and address. Consultant is an Employee. Programmer is an Employee. Each project has name, budget, starting date and ending date. Programming Language has name and platform. An Consultant may supervise many Projects. A Project can be supervised by only one Consultant. A Programmer works on at most two Projects. At least one Programmer works on a Project. A Programmer uses at least one Programming Language. A Programming Language is used by some number of Programmers. In a Project exactly one Programming Language is used. A Programming Language can be used by any number of Projects.

PROBLEM 3

A person is either a patient or a medical worker. A medical worker is either a doctor or a nurse. Medical workers work in a medical facility. A facility has a name, address, possibly a specialty area, and the name of an administrator. A patient visits a medical facility for a diagnosis of a health problem. A patient has name, id number (ssn), address (street, city, state, and zip), phone (day and evening), employer (company) name, employer address, type of benefits eligible for, and method for payment. Doctors are allowed to perform any kind of diagnosis and treatment based on their specialty. Nurses participate in treatment. Each medial worker must have at least one and possibly more assignments at a facility. Medical workers have certain skills that must be recorded and accessed for a new assignment.

PROBLEM 4

A company is organized into departments. Each department has a unique number, name and a manager. Each manager has a start date. A department may have several locations. A department controls a number of projects. Each project has a unique name, a unique number and a single location. We store each employee's name, social security number, address, salary, gender and birth date. An employee is assigned to one department. However each employee may work on several projects, which are not necessarily controlled by the same department. Each project needs some parts supplied by some suppliers. Each supplier has a unique name, a contact person's first and last name and an address. A part has a unique part number, a description and a cost.

PROBLEM 5

A concert season schedules one or more concerts. A concert season is identified by its opening date, which includes month, day, and year. A concert includes one or more compositions. A concert is identified by its number. A concert also has a concert date, which consists of month, day, year, and time. For each concert there is one conductor. A conductor is identified by his/her PID. A conductor name also needs to be included in the database. Each composition may require zero or more soloists. A composition is identified by its CID, which consists of composer name and composition name. Compositions have multiple movements. A movement is identified by an MID, which includes movement name and movement number. A soloist is identified by his/her PID. A soloist name is also kept in the database.

PROBLEM 6

A library owns one or more copies of some books. Each customer can check out one or more books from libraries that he is a member of. A library has name, working hours, types (free or member only), maximum limit of books that can be checked out and a address which contains street number, street name, city, zip code. A customer has a name, year of birth, ssn, phone number and address. A book has a name, year of publication, publisher, ISBN,

price, one or more subjects (e.g., comic, science fiction, literature) and a target audience age range (e.g., 1-3 years, 2-10 years, 18 and over). Each check out transaction has a check out date, an expected return date and an actual return date.

PROBLEM 7

The company has 50 plants located in 40 states and approximately 100,000 employees. Each plant is divided into departments and further subdivided into work stations. There are 100 departments and 500 work stations in the company. In each department there is an on-line time clock at which employees report their arrival and departure. A work task is associated with one of 20 different job types. Each of the job types can be performed at each of the plants. During a given day an employee may perform more than one work task, each associated with a different job type, and each can be performed at different work station. Each work station has an on-line data entry device at which an employee reports activity on a work task. There are five worker unions represented in the company, and every employee belongs to exactly one union. Although the size of the company remains stable, about 20 percent of the employees leave each year and are replaced by new personnel.

PROBLEM 8

Cell phone service plans are provided by different wireless service providers. A customer has a name, one or more email addresses and an address. The address contains street number, street name, city, zip code and state. Each customer can be an individual OR a business customer. Every individual has a year of birth, social security number and an age. Each business customer has a contact person name and a web address. A wireless service provider has a name, a web address, a customer service toll free number, an email address and a mailing address. Any service plan has a title, monthly fee, anytime minutes, the price of each minute which exceeds specified minutes in the service plan, flags identifying that the plan includes caller id, call forwarding or call waiting. Any phone, associated with a service plan, is assigned to one customer. Any phone has an area code, a number, current balance, service start date and a flag identifying whether the amount is past due.

PROBLEM 9

A College is divided into several schools. Each school is administered by a dean. Each dean is a member of administrators (ADMINISTRATOR). Deans may teach a class (PROFESSOR). Administrators and professors are also Employees. Each school is composed of several departments. Each department belongs to only a single school. Each department offers several courses. Each department has many professors. One of those professors chairs the department. Only one of the professors can chair the department. Each professor may teach at most four classes. A professor may also be on a research contract. A student may enroll in several classes. Each student may enroll in up to six classes and each class may have up to 35 students. Each student has only a single major and associated with a single department. Each student has an advisor in his or her department. Each advisor counsels several students. An advisor is also a professor.

PROBLEM 10

Each person is identified by his/her SSN. Each person also has a name, address, driver's license number and birth date. A person can own one or more cars (the ownership is declared in the title deed for the car). In addition, the same car can be owned by more than one person. Cars are identified by their VIN (Vehicle Identification Number). Each car also has a registered plate number which consists of the state that issued it, the actual number (which might contain letters), and what type of plate (is it a vanity plate, if so what kind, etc.). Each car, to be driven, must possess a valid insurance policy. An insurance policy consists of the car VIN, the car registration number, an expiration date, and also lists the acceptable drivers (for the moment assume that only those people who are listed on the policy are eligible to drive it).

APPENDIX H

PRE-EXPERIMENT QUESTIONNAIRE

Please answer the following questions about your education and experience. Your answers will be kept confidential. Please *circle* the options fit you best.

1. What education level are you attained?
   a) some college (what year?)
   b) bachelors
   c) masters level
   d) above masters level

2. What is your major?
   a) computer Sciences
   b) information Sciences
   c) others but taken database and other computer related courses
   d) others

3. In order to participate in this study, you stated that you have had some experience with or exposure to entity relationship diagrams for database design. Please indicate the type of experience you have with this type of diagram.
   a) did ERD exercises in database design classes some years ago
   b) constructed some small entity relationship diagrams for a class assignment or project recently

c) extensive experience of using ERD for database design

d) Expert knowledge of using ERD for database design

4. What software do you use when you draw ERD?

   a) Microsoft Visio

   b) Microsoft Paint

   c) Gnome Dia

   d) Others (Please name: _____)

APPENDIX I

POST-EXPERIMENT QUESTIONNAIRE

Please answer the questions about the experiment you just completed. We are interested in your opinions about the system. Your answers will be kept confidential. Please *circle* the answer that best corresponds with your opinions.

1. How hard is the problems?

	Problem 1	problem 2	problem 3	problem 4
a) very easy				
b) somewhat easy				
c) somewhat hard				
d) very hard				

2. How confident do you feel that your ERD solutions to problems are correct?

	Problem 1	problem 2	problem 3	problem 4
a) not confident				
b) somewhat not confident				
c) somewhat confident				
d) very confident				

3. Do you think the automated conceptual data modeling system helpful?
   a) not helpful at all
   b) not very helpful
   c) somewhat helpful
   d) very helpful

4. Do you think the automated conceptual data modeling system easy to use and user friendly?
   a) not at all
   b) not very easy to use
   c) somewhat easy to use and user friendly
   d) very easy to use and user friendly

5. What are the major advantages of using the automated conceptual data modeling tool?

6. What are the major disadvantages of using the automated conceptual data modeling tool?

7. Did the tool used prevent you from working efficiently?

8. Do you have any further comments or suggestions that you want to make? Please write them here.

BIBLIOGRAPHY

[Abb83] R. Abbott. Program design by informal english descriptions. *Commun. ACM 26(11):882-894*, 1983.

[ABM+94] D.J. Arnold, L. Balkan, S. Meijer, R. Humphreys, and L. Sadler. *Machine Translation: an Introductory Guide*. Blackwells-NCC, London, 1994.

[ACRG91] H. Alshawi, D.M. Carter, M. Rayer, and B. Gamback. Translation by quasi logical form transfer. In *the 29th ACL*, 1991.

[AG97] V. Ambriola and V. Gervasi. Processing natural language requirements. In *The 12th IEEE International Automated Software*, 1997.

[AG06] Vincenzo Ambriola and V. Gervasi. On the systematic analysis of natural language requirements with circe. *Automated Software Engineering, 13, 107-167*, 2006.

[All95] James Allen. *Natural Language Understanding (2nd Edition)*. Addison Wesley, 1995.

[AS88] Geert Andriaens and S.L. Small. Word expert revisited in a cognitive science perspective. In S. Small, G. Cottrell, and M. Tanenhaus, editors, *Lexical Ambiguity Resolution: Perspectives from Psycholinguistics, Neuropsychology, and Artificial Intelligence*, volume 13-43. Morgan Kaufman, 1988.

[BBLON77] J. Bubenko, S. Berlid, E. Lindencrona-Ohlin, and S.A. Nachmens. From information requirements to dbtg data structure. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1977.

[BCD+95] Edith Buchholz, H. Cyriaks, A. Dsterhft, H. Mehlan, and B. Thalheim. Applying a natural language dialogue tool for designing databases. In *Proceedings of the First International Workshop on Applications of Natural Language to Databases (NLDB'95)*, 1995.

[BD94] Edith Buchholz and A. Dsterhft. Using natural language for database design. In *Working Notes of the KI'94 Workshop: Reasoning about structured*

Objects: Knowledge Representation Meets Databases (KRDB-94), Saarbrueken, Germany, 1994., 1994.

[BDDL84] C. Batini, B. Demo, and A. Di Leva. A methodology for conceptual schema design of office databases. *Information Syst., Vol 9, 251-263*, 1984.

[BDT97] Edith. Buchholz, A. Diisterhoft, and B. Thalheim. Capturing information on behavior with the radd-nli: Linguistic and knowledge-based approach. *Data and Knowledge Engineering, 23:33-46*, 1997.

[BGM85] Mokrane Bouzeghoub, G. Gardarin, and E. Metais. Database design tools: An expert system approach. In *Very Large Data Base Conference*, 1985.

[Bla87] William J. Black. Acquisition of conceptual data models from natural language descriptions. In *3rd Conf. of the European chapter of ACM, Danemark*, 1987.

[BMO01] Bernhard Bauer, J.P. Muller, and J. Odell. Agent uml: A formalism for specifying multiagent interaction. *Agent-Oriented Software Engineering*, 2001.

[BMSW97] Daniel Bikel, S. Miller, R. Schwartz, and R. Weischedel. A high-performance learning name-finder. In *Fifth Applied Natural Language Processing Conference*, 1997.

[Boo86] G. Booch. Object-oriented development. *Transactions on Software Engineering, 12(2), 211-221*, 1986.

[BPPM91] Peter F. Brown, S. Pietra, V. Pietra, and R. Mercer. Word sense disambiguation using statistical methods. In *The 29th Annual Meeting on Association for Computational Linguistics, 264-270*, 1991.

[BR94] Eric Brill and P. Resnik. A rule based approach to prepositional phrase attachment disambiguation. In *Proceedings of the Fifteenth International Conference on Computational Linguistics (COLING)*, 1994.

[BRJ96] Grady Booch, J. Rumbaugh, and I. Jacobson. The unified modeling langauge for object-oriented development. *Unix Review*, 1996.

[BSA72] I.D.J. Bross, P.A. Shapiro, and B.B. Anderson. How information is carried in scientific sublanguages. *Science, 176:1303-1307*, 1972.

[BvdR96] J.F.M. Burg and R.P. van de Riet. A natural language and scenario based approach to requirements engineering. In *In Natuerlichsprachlicher Entwurf von Informationssystemen (NEI'96), Tutzing, Germany*, 1996.

[CB95] Michael Collins and J. Brooks. Prepositional phrase attachment through a backed of model. In *Proceedings of the Third Workshop on Very Large Corpora,27-38*, 1995.

[CGG92] Jim Cowie, J. Guthrie, and L. Guthrie. Lexical disambiguation using simulated annealing. In *the 14th International Conference on Computational Linguistics, Vol. 1. 359-365*, 1992.

[Cha96] Eugene Charniak. Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, pages 1031-1036, Menlo Park. AAAI Press/MIT Press.*, 1996.

[Che76] Peter Chen. The entity relationship model toward an unified view of data. *ACM Transactions on Database Systems (TODS)*, 1976.

[Che81] Peter Chen. A preliminary framework for entity-relationship model. *Enity-Relationship Approach to Information Modeling and Analysis*, 1981.

[Che83] Peter Chen. English sentence structure and entity relationship diagrams. *Information Science Vol.1, No. 1, Elsevier (1983) 127-149*, 1983.

[Chr98] Fellbaum Christiane. *WordNet: an electronic lexical database.* Language, speech, and communication. MIT Press, Cambridge, Mass, 1998.

[CM00] Aone Chinatsu and R.S. Mila. Rees: a large-scale relation and event extraction system. In *Proceedings of the sixth conference on Applied natural language processing*, 2000.

[CMNK88] J. Choobineh, M. Mannino, J.F. Nunamaker, and B. Konsynsky. An expert database design system based on analysis of forms. *IEEE Transaction on Software Engineering, Volume 14, 242-253*, 1988.

[Cos97] Marco Costantino. Financial information extraction using pre-defined and user-definable templates in the lolita system. *Journal of Computing and Information Technology, Vol. 4 No.4*, 1997.

[Cyr95] Walling Cyre. A requirements sublanguage for automated analysis. *CyrW95a] International Journal of Intelligent Systems, 10 (7), 665-689, July 1995.*, 1995.

[Dal92] H. Dalianis. A method for validating a conceptual model by natural language discourse generation. In *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering*, 1992.

[DI90] Ido Dagan and A. Itai. Automatic processing of large corpora for the resolution of anaphora references. In *the 13th International Conference on Computational Linguistics (COLING'90)*, 1990.

[DM06] Siqing Du and D.P. Metzler. An automated multi-component approach to extract entity relationship from database requirement specification document. In *11th International Conference on Applications of Natural Language to Information Systems, Austria*, 2006.

[dMMM06] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *LREC*, 2006.

[DP95] Eric Dubois and M. Petit. Using a formal declarative language for specifying requirements modelled in cimosa. In *Proc. of the European workshop on Integrated Manufacturing Systems Engineering – IMSE'94*, 1995.

[DPL94] Ido Dagan, Fernando C. N. Pereira, and Lillian Lee. Similarity-based estimation of word cooccurrence probabilities. In *Meeting of the Association for Computational Linguistics*, pages 272–278, 1994.

[DZBG96] Walter Daelemans, J. Zavrel, P. Berck, and S. Gillis. Mbt: A memory-based part of speech tagger-generator. In *Proceedings of the Fourth Workshop on Very Large Corpora, Copenhagen, Denmark, 14-27*, 1996.

[EDB00] Letha H. Etzkom, C.G. Davis, and L.L. Bowen. The language of comments in computer software: A sublanguage of english. *Journal of Pragmatics 33 (2001) 1731-1756*, 2000.

[EGK+04] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz and dynagraph — static and dynamic graph drawing tools. In Michael Junger and Petra Mutzel, editors, *Graph Drawing Software*, Mathematics and Visualization, pages 127–148, Berlin-Heidelberg-New York-Hong Kong-London-Milan-Paris-Tokyo, 2004. Springer-Verlag.

[EJ00] de Jong Edwin and van de Pol Jaco. Refinement in requirements specification and analysis: A case study. In *the 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, 2000.

[EL85] C.F. Eick and P.C. Lockemann. Acquisition of terminology knowledge using database design techniques. *Proceedings ACM SIGMOD Conference, Austin, USA, 84-94*, 1985.

[EL89] David W. Embley and T.W. Ling. Synergistic database design with an extended entity-relationship model. *ER 1989, pp. 111-128*, 1989.

[FGR+94] A. Fantechi, S. Gnesi, G. Ristori, M. Carinin, M. Vanocchi, and P. Moreschini. Assisting requirement formalization by means of natural language translation. *Formal Methods in System Design*, 1994.

[Fie98] Glenn Fields. Proposal for a national consortium on controlled languages and computer-aided translation. In *the National Consortium to Advance Controlled Language and Computer-Aided Translation Tools (NCCAT) Kick-Off Meeting*, 1998.

[Fil68] Charles J. Fillmore. The case for case. In Harms Bach, editor, *Universals in Linguistic Theory*. New York: Holt, Rinehart, and Winston, 1-88., 1968.

[FKM+96] G. Fliedl, Ch. Kop, W. Mayerthaler, H.C. Mayr, and Ch. Winkler. Nts based derivation of kcpm cardinalities: From natural language to conceptual predesign. *Application of Natural Language to Information Systems*, 1996.

[FKM+98] G. Fliedl, C. Kop, W. Mayerthaler, H.C. Mayr, and C. Winkler. Disambiguation of part of relationships within the niba project. In *DEXA98 Workshop Proceedings, Vienna, 171 - 175*, 1998.

[Fou99] Alain Jrme Fougres. Formal specifications building from specifications written in natural language. *Computational Linguistics,*, 1999.

[FS96] N.E. Fuchs and R. Schwitter. Attempto controlled english (ace). In *Proceeding of CLAW96, the 1st International Workshop Controlled Language Applications*, 1996.

[Fur04] Johannes Furnkranz. Web mining. In Oded Maimo and L. Rokach, editors, *The Data Mining and Knowledge Discovery Handbook, 899– 920. Springer,2005.* Springer, 2004.

[Gal02] Alexandra Galatescu. Reifying model integration abilities from natural language. *Journal of Conceptual Modeling*, 2002.

[GB97] L. Goldin and D.M. Berry. Abstfinder, a prototype natural language text abstraction finder for use in requirements elicitation. *Automated Software Engineering 4(4):375-412*, 1997.

[GLS95] Dennis Grinberg, J. Lafferty, and D. Sleator. A robust parsing algorithm for link grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies, Prague, September, 1995.*, 1995.

[GN00] Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *Software — Practice and Experience*, 30(11):1203–1233, 2000.

[GN02] Vincenzo Gervasi and B. Nuseibeh. Lightweight validation of natural language requirements. *Software: Practice and Experience Volume 32, Issue 2. Pages 113-133*, 2002.

[GR98] J. Giarratano and G. Riley. *Expert Systems: Principles and Programming (3 ed.)*. Boston: PWS Publishing, 1998.

[Gra] Jeff Gray. Collection of ambiguous or inconsistent/incomplete statements.

[Gri97] Ralph Grishman. Information extraction: Techniques and challenges. In *SCIE*, pages 10–27, 1997.

[Gri01] Ralph Grishman. Adaptive information extraction and sublanguage analysis. In *In Proceedings of the Workshop on Adaptive Text Extraction and Mining at the 17 International Joint Conference on Artificial Intelligence*, 2001.

[Gro77] Barbara Jean Grosz. The representation and use of focus in a system for understanding dialogs. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1977.

[GSD99] Fernando Gomez, C. Segami, and C. Delaune. A system for the semiautomatic generation of er models from natural language specifications. *Data and Knowledge Engineering 29 (1) 57-81*, 1999.

[GSH97] F. Gomez, C. Segami, and R. Hull. Determining propositional attachment, propositional meaning, verb meaning and thematic roles. *Computational Intelligence 3(1) 1-31*, 1997.

[HAB+97] Jerry R. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. Fastus: A cascaded finite state transducer for extracting information from natural language text. In *Finite-State Devices for Natural Language Processing, pp. 383-406, MIT Press, Cambridge, MA*, 1997.

[Har89] Zellig S. Harris. *The Form of information in science: analysis of an immunology sublanguage.* Dordrecht [Netherlands] and Boston: Kluwer Academic Publishers, 1989.

[Hay78] Philip J. Hayes. Mapping input into schemas. *Technical report 29, Department of Computer Science, University of Rochester*, 1978.

[HBR03] V.M. Harsha, N. Balakrishnan, and K.R. Ramakrishnan. Event information extraction using link grammar. In *13th International WorkShop on Research Issues in Data Engineering: Multi-lingual Information Management (RIDE'03).*, 2003.

[Hea91] Marti A. Hearst. Noun homograph disambiguation using local context in large corpora. In *Proceedings of the 7th Annual Conference of the University of Waterloo Centre for New OED and Text Research, 1-19*, 1991.

[HG03] H.M. Harmain and R. Gaizauskas. Cm-builder: A natural language-based case tool for object-oriented analysis. *Automated Software Engineering 10 (2) 157-181*, 2003.

[Hir81] Graeme Hirst. Discourse-oriented anaphora resolution in natural language understanding: A review. *American Journal of Computational Linguistics*, 1981.

[HJL96] Constance L. Heitmeyer, R.D. Jeffords, and B.G. Labaw. Automated consistency checking of requirements specifications. *ACM Transactions on Software Engineering and Methodology*, 1996.

[Hud84] Richard A. Hudson. *Word Grammar*. Basil Blackwell, Oxford, 1984.

[IS89] R. Ingria and D. Stallard. A computational mechanism for pronominal reference. In *the 27th Annual Meeting of the ACL, 262-271*, 1989.

[Isa84] P. Isabelle. Machine translation at the taum group. *The ISSCO Tutorial on Machine Translation*, 1984.

[IV98] Nancy Ide and J. Veronis. Word sense disambiguation:the state of the art. *Computational Linguistics, 24:1, 1-40.*, 1998.

[JM00] Dan Jurafsky and J.H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition.* Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River, N.J., 2000.

[KAMN98] Christine Kamprath, E. Adolphson, T. Mitamura, and E. Nyberg. Controlled language multilingual document production: Experience with caterpillar technical english. In *In Proceedings of the Second International Workshop on Controlled Language Applications (CLAW98)*, 1998.

[Kit87] R.I. Kittredge. *The Significance of Sublangauge for Automatic translation.* Machine Translation: Theoretical and Methodological Issues. Cambridge University Press, 1987.

[KM95] Y. G. Kim and S. T. March. Comparing data modeling formalisms. *Communications of the ACM, Vol. 38, No. 6, pp. 103-115*, 1995.

[KM02] Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 3–10. MIT Press, 2002.

[Kob99] Cris Kobryn. Uml2001: A standardization odyssey. *Commun. ACM 42(10):29-37*, 1999.

[LB02a] Beum-Seuk Lee and B.R. Bryant. Automated conversion from requirements documentation to an object-oriented formal specification language. *Proceedings of the 2002 ACM symposium on Applied computing table of contents Pages: 932 - 936*, 2002.

[LB02b] Beum-Seuk Lee and B.R. Bryant. Automation of software system development using natural language processing and two-level grammar. In *Proceedings of Monterey Workshop*, 2002.

[LB02c] Beum-Seuk Lee and B.R. Bryant. Contextual natural language processing and daml for understanding software requirements specifications. In *Proceedings of the 19th International Conference on Computational Linguistics, 516-522*, 2002.

[Lee03] Beum-Seuk Lee. *Automation of Software System Development Using Natural Language Processing and Two-Level Grammar*. PhD thesis, UAB, 2003.

[LG90] B.Douglas Lenat and R.V. Guha. *Building large knowledge-based systems: representation and inference in the Cyc project.* Addison-Wesley Publishing Company, Inc., 1990.

[Lin98] D. Lin. Dependency-based evaluation of minipar. In *In Workshop on the Evaluation of Parsing Systems*, 1998.

[LL94] Shalom Lappin and H. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics, 20(4), 535-561*, 1994.

[LRW01] G. Leech, P. Rayson, and A. Wilson. *Word Frequencies in Written and Spoken English: Based on the British National Corpus*. Pearson ESL, 2001.

[LS04] Hugo Liu and P. Singh. Conceptnet: A practical commonsense reasoning toolkit. *BT Technology Journal.*, 2004.

[Mar87] M. Marcus. Deterministic parsing and description theory. In P. Whitelock, M. Wood, H. Somers, R. Johnson, and P. Benett, editors, *Linguistic Theory and Computer Applications.* Academic Press, 1987.

[MFNI04] Luisa Mich, M. Franch, and P. Novi Inverardi. Market research for requirements analysis using linguistic tools. *Requirements Eng:40-56*, 2004.

[MHF83] M. Marcus, D. Hindle, and M. Fleck. D-theory: Talking about talking about trees. *Proceeding of Association for Computational Linguistics.*, 1983.

[Mic96] L Mich. Nl-oops: From natural language to object oriented requirements using the natural language. *Journal of Natural Language Engineering 2(2) 161-187*, 1996.

[Mit98] Ruslan Mitkov. Robust pronoun resolution with limited knowledge. In *(COLING'98), the 18th International Conference on Computational Linguistics*, 1998.

[Mit99] Ruslan Mitkov. Anaphora resolution: the state of the art. In *Working paper, University of Wolverhampton*, 1999.

[MMBS93] Mitchell P. Marcus, M.A. Marcinkiewicz, and B. Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *computational Linguistics Volume 19 (2)*, 1993.

[MMS+02] PalakalR Mathew, S. Matthew, M. Snehasis, R. Rajeev, and R. Simon. A multi-level text mining method to extract biological relationships. In *IEEE Computer Society Bioinformatics Conference*, 2002.

[Mor93] T. Morimoto. Atr's speech translation system: Asura. In *Proc. 3rd European Conf. on Speech Communication and Technology, pp. 1291–1294*, 1993.

[Mor97] A. M. Moreno. Object-oriented analysis from textual specifications. In *In Proceedings of Ninth International Conference on Software Engineering and Knowledge Engineering, Madrid, Spain*, 1997.

[MP06] Ryan T. McDonald and Fernando C. N. Pereira. Online learning of approximate dependency parsing algorithms. In *EACL*. The Association for Computer Linguistics, 2006.

[NDE03] Pierre Nugues, S. Dupuy, and A. Egges. Information extraction to generate visual simulations of car accidents from written descriptions. In *Lecture Notes in Computer Science: Computational Science and Its Applications-ICCSA*, 2003.

[NL03] C.J. Neill and P.A. Laplante. Requirement engineering: the state of the practice. *IEEE Software 20(6), 40-45*, 2003.

[NN05] Joakim Nivre and Jens Nilsson. Pseudo-projective dependency parsing. In *ACL*. The Association for Computer Linguistics, 2005.

[NR95] Sastry Nanduri and S. Rugaber. Requirements validation via automated natural language parsing. *Proceedings of the 28th Annual Hawaii International Conference on System Sciences - 1995*, 1995.

[OHM04] Nazlia Omar, P. Hanna, and P. McKevitt. Heuristics-based entity-relationship modelling through natural language processing. In *Proc. of the Fifteenth Irish Conference on Artificial Intelligence and Cognitive Science (AICS-04), Lorraine McGinty and Brian Crean (Eds.), 302-313*, 2004.

[OM96] Miles Osborne and C.K. MacNish. Processing natural language software requirement specifications. In *Second International Conference on Requirements Engineering (ICRE'96)*, 1996.

[PAC+93] S.G. Pulman, H. Alshawi, D. Carter, R. Crouch, M. Rayner, and A. Smith. Clare: a combined language and reasoning engine. In *Technical Report CRC-042, SRI International, Cambridge, Mass.*, 1993.

[Qui67] M.Ross Quilian. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science, 12, 410-30*, 1967.

[RBA98] C. Rolland and C. Ben Achour. Guiding the construction of textual use case specifications. *Data and Knowledge Engineering Journal Vol. 25 No. 1-2, 125-160.*, 1998.

[RDH+02] Fabio Rinaldi, J. Dowdall, M. Hess, D. Moll, and R. Schwitter. Towards answer extraction: An application to technical domains. In *Proceedings of the 15th European Conference on Artificial Intelligence, IOS Press, Amsterdam*, 2002.

[Rol88] Colette Rolland. *An Information System Methodology supported by an Expert Design Tool*. University of Paris, Elsevier Science, 1988.

[RP92] C. Rolland and C. Prox. A natural language approach for requirements engineering. *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering*, 1992.

[RP00] C. Rolland and N. Prakash. From conceptual modelling to requirements engineering. *Annals of Software Engineering, 10:151–176*, 2000.

[RRR94] Adwait Ratnaparkhi, J. Reynar, and S. Roukos. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the Human Language Technology Workshop, 250-255*, 1994.

[RW91] Howard B. Reubenstein and R.C. Waters. The requirements apprentice: Automated assistance for requirements acquisition. *IEEE Transactions on Software Engineering, Vol 17*, 1991.

[SC04] William Scott and S. Cook. A requirements assessment architecture that combines natural language parsing and artificial intelligence. *INCOSE 2004 - 14th Annual International Symposium Proceedings*, 2004.

[Sch98] Gerold Schneider. *A Linguistic Comparison of Constituency, Dependency and Link Grammar*. PhD thesis, 1998.

[SFL87] Naomi Sager, C. Friedman, and M.S. Lyman. *Medical Language Processing, Computer Management of Narrative Data*. Addison-Wesley Publishing Company, Inc., 1987.

[SHE89] Motoshi Saeki, H. Horai, and H. Enomoto. Software development process from natural language specification. In *Proceedings of the 11th International Conference on Software engineering*, 1989.

[SHT+87] Motoshi Saeki, H. Horai, K. Toyama, N. Uematsu, and H. Enomoto. Specification framework based on natural language. In *Proceeding of 4th International Workshop on Software Specification and Design, 87-94*, 1987.

[Sid78] Candace Lee Sidner. A progress report on the discourse and reference components of pal. In *Proceedings of the Second National Conference, Canadian Society for Computational Studies of Intelligence*, 1978.

[SM00] Stephanie Strassel and N. Martey. Tdt2 careful transcription text, 2000.

[SN97] Jiri Stetina and M. Nagao. Corpus based pp attachment ambiguity resolution with a semantic dictionary. In *Proceedings of WVLC'97, 66-80*, 1997.

[Sow84] John F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley Publishing Company, Reading, MA, 1984.

[Sow91] John F. Sowa. *Principles of Semantic Networks: Explorations in the representation of Knowledge*. Morgan Kaufmann, 1991.

[ST91] K.Daniel Sleator and D. Temperley. Parsing english with a link grammar. In *Third International Workshop on Parsing Technologies*, 1991.

[Sto02] Veda C. Storey. Common sense reasoning in automated database design: an empirical test. *Journal of Database Management*, 2002.

[Sus93] Michael Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of the Second International Conference on Information and Knowledge Base Management, CIKM'93, Arlington, Virginia, 67-74.*, 1993.

[SWL83] Nan C. Shu, H. Wong, and V. Lum. Forms approach to requirements specification for database design. In *Proceedings of the 1983 ACM SIGMOD International Conference on Management of Data.*, 1983.

[TB93] A. Min Tjoa and L. Berger. Transformations of requirements specifications expressed in natural language into an eer model. In *Proceedings of the 12th International Conference on the Entity-Relationship Approach: Entity-Relationship Approach*, 1993.

[TF80] Toby J. Teorey and J.P. Fry. The logical record access approach to database design. *ACM Compute Surveys 12:2*, 1980.

[TF82] Toby J. Teorey and J.P. Fry. *Design of Database Structures*. Prentice-hall, Inc., 1982.

[TJS91] J.P. Tsai, H.C. Jang, and K.J. Schellinger. Rt-frorl: a formal requirements specification language for specifying real-time systems. In *Proceedings of the Fifteenth Annual International Computer Software and Applications Conference, COMPSAC '91.,*, 1991.

[vR79] C.J. van Rijsbergen. *Information Retrieval, second edition*. London, 1979.

[WR82] Peter C.S. Wong and E.R. Reid. Flair-user interface dialog design tool. *Acm SIGGRAPH Computer Graphics*, 16(3), 1982.

[WRA01] WRAIR. Cara specification: Proprietary document. *Walter Reed Army Institute for Research Technical Report, WRAIR, Dept. of Resuscitative Medicine*, 2001.

[WRH97] W.M. Wilson, L.H. Rosenberg, and L.E. Hyatt. Automated analysis of requirement specifications. In *19th International Conference on Software Engineering*, 1997.

[Yar92] David Yarowsky. Word sense disambiguation using statistical models of roget's categories trained on large corpora. *the 14th International Conference on Computational Linguistics, 454-460*, 1992.

[Yar95] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics, 189-196*, 1995.

[ZD99] Jakub Zavrel and W. Daelemans. Recent advances in memory-based part-of-speech tagging. In *Actas del VI Simposio Internacional de Comunicacion Social, Santiago de Cuba, pp. 590-597, 1999. ILK pub: ILK-9903.*, 1999.

[ZDJV97] Jakub Zavrel, W. Daelemans, and J. Jorn Veenstra. Resolving pp attachment ambiguities with memory-based learning. *T.M. EUison (ed.) CoNLL97: Computational Natural Language Learning, ACL,136-144.*, 1997.