

The Power of Constraint Grammars Revisited

Anssi Yli-Jyrä

University of Helsinki, Finland
anssi.yli-jyra@helsinki.fi

Abstract

Sequential Constraint Grammar (SCG) (Karlsson, 1990) and its extensions have lacked clear connections to formal language theory. The purpose of this article is to lay a foundation for these connections by simplifying the definition of strings processed by the grammar and by showing that Nonmonotonic SCG is undecidable and that derivations similar to the Generative Phonology exist. The current investigations propose resource bounds that restrict the generative power of SCG to a subset of context sensitive languages and present a strong finite-state condition for grammars as wholes. We show that a grammar is equivalent to a finite-state transducer if it is implemented with a Turing machine that runs in $o(n \log n)$ time. This condition opens new finite-state hypotheses and avenues for deeper analysis of SCG instances in the way inspired by Finite-State Phonology.

1 Introduction

Lindberg and Eineborg (1998), Lager and Nivre (2001) and Listenmaa (2016) have analyzed the Sequential Constraint Grammar (SCG) (Karlsson, 1990) from the logical point of view, proposing that the rules can be expressed in first-order Horn clauses, first-order predicate logic or propositional logic. However, many first-order logical formalisms are themselves quite expressive as Horn-clauses are only semi-decidable and first-order logic is undecidable, thus at least as powerful as SCG itself. Propositional logic is more restricted but does not help us to analyse the expressive power of SCGs and to prove the finiteness of grammars.

Instead of just reducing SCG to undecidable

or otherwise powerful formalisms, we are interested in the ultimate challenge that tries to prove that a *practical* grammar is actually reducible to a strictly weaker formalism. This goal is interesting because this kind of narrowing reductions have been proven extremely valuable. For example, the proof that practical grammars in Generative Phonology are actually equivalent to finite-state transducers has turned out to be a game-changing result. In fact, the reduction gave birth to the influential field of Finite-State Phonology.

It is noteworthy that prior efforts to analyse SCG in finite-state terms have focused on the finite-state nature of individual and parallel rules (Peltonen, 2011; Hulden, 2011; Yli-Jyrä, 2011). The efforts have mostly ignored the generative power of the grammar system as a whole and that of practical grammar instances.

In this paper, we are aiming to Finite-State Syntax through reductions of practical SCGs. To set the formal framework, we have to start, however, from the total opposite: we show first that the simplified formalism for Nonmonotonic SCGs is Turing equivalent and thus similar to Generative Phonology (Chomsky and Halle, 1968; Ristad, 1990) and Transformational Grammar (Chomsky, 1965; Peters and Ritchie, 1973). This foundational result gives access to the large body of literature of bounded Turing machines and especially to Hennie machines that run in $O(n)$ time and are equivalent to finite-state machines. Then the Gap Theorem (Trakhtenbrot, 1964) gives us access to a looser bound $o(n \log n)$ whose reasonable approximations are sufficient and decidable conditions for finite-state equivalence. We present some ways in which these bounds can be related to SCG parsing.

The article is structured as follows. Section 2 describes the alphabets, the strings and the derivation steps in SCG parsing. In Section 3, these are used to show Turing equivalence of SCGs. In next two sections, simple bounds are introduced and

elaborated further to obtain specific conditions for finite-state equivalence of grammars. Further links to formal language theory and two important open problems are presented in Section 6. Then the paper is concluded.

2 SCG as a "Phonological" Grammar

In the SCG literature, morphosyntactic readings of tokens are usually represented as tag strings like "<went>" "go" V PAST. The tag strings are now viewed as a compressed representation for a huge binary vector $(f_0, f_1, f_2, \dots, f_k, \dots)$. The semantics of the grammar ignores some tags and considers only k tags declared in advance in the grammar. These k tags or features distinguish readings from each other and define the *reading alphabet* $\Sigma = 2^k$.

An ambiguous token has more than one reading associated to it. The elements of the *cohort alphabet* $\mathcal{P}(\Sigma)$ are called cohorts. This alphabet is the powerset of the reading alphabet. Only a small subset of all possible cohorts occur in practice.

The input of an SCG is produced by a deterministic finite-state function, $Lexicon^* : T^* \rightarrow (\mathcal{P}(\Sigma))^*$, that maps token strings to *lexical cohort strings* of the same length. This function is the concatenation closure of the function $Lexicon : T \rightarrow (\mathcal{P}(\Sigma))$ that maps every token to a cohort.

Since the image of each token is a set of strings, $Lexicon$ is internally a nondeterministic lexical transducer (Karttunen, 1994; Chanod and Tapanainen, 1995), but the image of each token is viewed externally as a *symbol* in $\mathcal{P}(\Sigma)$, making $Lexicon$ a one-valued function.

An SCG processes the lexical cohort string by iterated application of a derivation step \Rightarrow : $(\mathcal{P}(\Sigma))^* \rightarrow (\mathcal{P}(\Sigma))^*$ that affects one cohort at a time. The contexts conditions of each derivation step are normally defined using an existing SCG formalism for contextual tests. Monadic Second Order Logic (Büchi, 1960; Elgot, 1961; Trakhtenbrot, 1961) provides an alternative formalism that can express all finite state languages over $\mathcal{P}(\Sigma)$.

The parser defines the parsing strategy that resolves the conflicts between rules that could be applied simultaneously. A typical strategy chooses always the most reliable rule and the leftmost target position. When the plain contextual tests are combined with the application strategy, we obtain a total functional transducer (Skut et al., 2004; Yli-Jyrä, 2008; Hulden, 2009). E.g., the transducer in

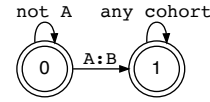


Figure 1: A simple \Rightarrow relation as an FST

Fig. 1 is total and replaces A by B in the first possible occurrence position.

The semantics of an SCG grammar G is defined as the relation

$$[[G]] = \{(i, o) \mid i \in \mathcal{P}(\Sigma)^*, o \in I, i \Rightarrow^* o\}$$

where $I \subseteq \mathcal{P}(\Sigma)^*$ as $\{x \mid (x, x) \in \Rightarrow\}$. This semantics makes SCG grammars similar to grammars in Generative Phonology (Chomsky and Halle, 1968) as both grammars relate the lexical string into some kind of output string by applying a sequence of alternation rules.

3 Nonmonotonic SCG

Two recent SCG implementations (Tapanainen, 1996; Didriksen, 2017) are *nonmonotonic*: they do not always reduce the input but they can insert tags, readings and even cohorts. In this section, we study the expressive power of such SCGs.

3.1 Minimal Definition

For the sake of minimality, we define the Non-monotonic SCG (NM-SCG) as a rule system that supports the following kinds of local transformation rules:

- REPLACE (*old*) (*new*) (*cond*)⁺
- INSCOHORT (*targ*) (*cond*)⁺
- REMCOHORT (*targ*) (*cond*)⁺

The first rule template in the above replaces the leftmost cohort containing the reading *old* with a cohort that contains the reading *new* if the relative context condition *cond* is satisfied. The familiar SELECT and DELETE rules are seen as shorthands for sets of REPLACE rules. The second and the third rule templates are used to insert or remove a target cohort matching the pattern *targ* when the condition *cond* is satisfied. The plus (⁺) indicates that more than one condition can be present.

Our simplified context conditions are of the form (*d tags*) or (*d NOT tags*) where the first tests the presence of the pattern *tags* in the relative cohort location *d*. The second is true when the location does not contain the pattern.

3.2 One-Tape Turing Machine

A one-tape deterministic *Turing machine* (TM) has a finite control unit and an infinite rewritable tape with a pointer (Fig. 2). A configuration of the machine consists of the current state q , the current pointer value and the contents of the working tape.

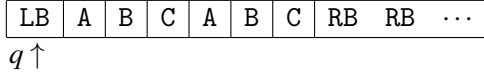


Figure 2: A one-tape Turing machine

The tape is divided into squares that hold a left boundary LB, a right boundary RB, or a symbol from the tape alphabet Ω . Given the input string $x \in \Omega^*$, the first square of the tape is pointed and the tape is initialized with the prefix LBxRB that is followed by an infinite number of right boundary symbols.

The control unit is a deterministic finite automaton where each transition $s \xrightarrow{(A,B,d)} t$ specifies the source state s , the target state t , the input symbol A , the output symbol B , and a head move $d \in \{-1, 0, 1\}$. On each transition, the machine overwrites the symbol A in the pointed square with the symbol B , changes its state from s to t and then moves the pointer d steps to the right. All but the leftmost square are over-writable ($B = A$ if $A = LB$), but the machine never moves beyond the first right boundary without overwriting it with a tape symbol and never writes RB between two tape symbols.

The computation of the machine starts from state q_0 . At each step, the machine takes the next transition based on the current state and the currently pointed symbol on the memory tape. The computation continues as long as the next transition is defined and then halts by reaching a state from which there is no transition on the current input. If the halting state is among the final states F , the machine accepts the input contents and relates it with the string $x' \in \Omega$ stored to the memory tape. Otherwise, the machine either gets stuck to an infinite computation or gives up, leaving some ill-formed string to the memory tape.

3.3 Reduction to Nonmonotonic SCG

Now we show that any one-tape Turing machine can be simulated with a nonmonotonic SCG.

In our simulation, each square in the initial portion of the memory tape corresponds to a cohort in

the input. Each cohort is a singleton set in $\mathcal{P}(\Sigma)$ i.e. represents just one reading in Σ . Each reading is a collection of positive features from Φ . These features include the tape symbols Ω , the boundary symbols $\{LB, RB\}$, and the markers that we need to keep track of the computation steps.

The pointed square corresponds to a cohort that contains a marker. Since SCG can change only one cohort at a time, movement of the pointer involves two temporarily marked positions and markers: the first indicates the previously pointed square and the second indicates the new pointed square. One marker represent the source state and the other represents the transition in progress.

A transition $q \xrightarrow{A,B,d} r$, $RB \notin \{A, B\}$, corresponds to a sequence of three rule applications that change one cohort at a time. Since the set of transitions, the sets of states Q and the tape alphabet Ω are finite, each step is described with a finite set of non-monotonic SCG rules:

1. Given the state marker $Q_q \in \{Q_s \mid s \in Q\} \subseteq \Phi$ in cohort i and no other marked cohorts, add a transition marker $T-q-A \in \Phi$ to cohort $i+d$ that previously contains a tape symbol $C \in \Omega$:

$$\text{REPLACE } (C) (T-q-A \ C) (-d \ Q_q \ A)$$

2. Given a transition marker $T-q-A$ in cohort $i+d$, overwrite, in cohort i , the reading containing the tape symbol A and the state marker Q_q with a reading containing the tape symbol B :

$$\text{REPLACE } (Q_q \ A) (B) (d \ T-q-A)$$

3. When no state marker is present, replace the transition marker $T-q-A$ with the marker for the target state Q_r while keeping the remainder $C \in \Sigma$ in the changed cohort:

$$\text{REPLACE } (T-q-A \ C) (Q_r \ C) (-d \ \text{NOT } Q_q)$$

A transition $q \xrightarrow{RB,A,0} r$, $A \in \Omega$, corresponds to the application of rules:

$$\begin{aligned} &\text{ADDCOHORT } (T-q-RB \ A) (1 \ Q_q \ RB) \\ &\text{REPLACE } (Q_q \ RB) (RB) (-1 \ T-q-RB \ A) \\ &\text{REPLACE } (T-q-RB \ A) (Q_r \ A) (1 \ \text{NOT } Q_q \ RB) \end{aligned}$$

When the previous cohort contains tape symbol $C \in \Omega$, a transition $q \xrightarrow{A,RB,-1} r$, where $A \in \Omega$, corresponds to the application of rules:

$$\begin{aligned} &\text{REPLACE } (C) (T-q-A \ C) (1 \ Q_q \ A) (2 \ RB) \\ &\quad \text{REMOHORT } (Q_q \ A) (1 \ T-q-A) \\ &\text{REPLACE } (T-q-A \ C) (Q_r \ C) (1 \ \text{NOT } Q_q \ A) \end{aligned}$$

Transitions $q \xrightarrow{\text{RB},A,-1} r$, $q \xrightarrow{\text{RB},A,1} r$ and $q \xrightarrow{A,\text{RB},0} r$, $A \in \Omega$, reduce to a sequence of two transitions.

The SCG parser halts when the tape contents does not trigger any of these rules that simulate transitions. The simulation accepts the input if some cohort contains a marker Q_q such that $q \in F$.

Proposition 1. *NM-SCGs can simulate TMs.*

Since NM-SCG is itself an algorithm, we have:

Proposition 2. *There is a one-tape deterministic TM that implements the NM-SCG parser.*

Proposition 3. *NM-SCGs are equivalent to TMs.*

4 Bounded Nonmonotonic SCGs

The undecidability of Nonmonotonic SCG creates a need to restrict the formalism in ways that ensure decidability. In this section, we propose two parameters that set important bounds on the resources available to grammars.

4.1 The $O(n)$ Space Bound

The *fertility* $f \in \mathbb{N} \cup \{\infty\}$ of a nonmonotonic SCG grammar is the maximum number of new cohorts that each the grammar inserts before any of the n cohorts in the original sentence (with RB). Note that fertility $f > 0$ implies nonmonotonicity.

Proposition 4. *In finite-fertility SCGs, the length ℓ of the output string is linearly bounded.*

The bounded length of the cohort string is an important restriction to Nonmonotonic SCGs because it ensures that any infinite loop in the computation can be detected after a bounded number of computation steps because the number of distinct tape contents is bounded.

Proposition 5. *The termination of a finite-fertility SCG is decidable.*

We also know that the preconditions of each rule can be tested with a finite automaton and that the actual effect on the target cohort is a functional finite-state computation that can be implemented in linear space according to the length of the cohort string.

Proposition 6. *The space requirement of a finite-fertility SCG is linear to the maximum length of the cohort string during the derivation.*

A deterministic linear-bounded automaton (DLBA) (Myhill, 1960) is a special case of Turing machines with the restriction that the right boundary is fixed and cannot be overwritten. The

LBA computations can be initialized so that the space available for storing the cohort string is linearly bounded by the length of the initial cohort string.

Proposition 7. *A nonmonotonic SCG with finite fertility is simulated by an DLBA.*

The power of DLBAs is restricted to a strict subset of context-sensitive languages (Kuroda, 1964).

Proposition 8. *The cohort language accepted by a finite-fertility SCG is context sensitive.*

4.2 The $O(n^2)$ Time Bound

By studying only monotonic SCGs with the reading count r in cohorts, and the sentence length n (including RB), Tapanainen (1999) has given a lower bound for the parsing time:

Proposition 9 (Tapanainen 1999). *Any monotonic SCG performs $O(nr)$ rule applications.*

The *volume* $v \in \{1, 2, \dots\} \cup \{\infty\}$ of cohorts is a parameter that tells the maximum number of operations that can be applied to any cohort. This new notion is a nonmonotonic generalization of the maximum number of readings in one cohort. Finite volume basically turns every finite fertility SCG into a monotonic SCG.

Finite fertility helps us to generalize the above proposition to nonmonotonic SCGs.

Proposition 10. *Any NM-SCG performs $O((1+f)nv)$ rule applications.*

Assuming again that any rule of the grammar can be applied in linear time according to the number of cohorts, we obtain a time complexity result:

Proposition 11. *Any NM-SCG runs in $O((1+f)^2 n^2 v)$ time.*

5 Finite-State Hypotheses

A deterministic linear bounded automaton is a special case of one-tape deterministic Turing machines that gives us a context where many interesting conditions for finite-stateness aka regularity become applicable.

5.1 The $o(n \log n)$ Time Bound

Hennie (1965) showed that a deterministic one-tape TM running in $O(n)$ is equivalent to a finite automaton. By defining the relation between the initial and final tape contents, we can extend Hennie's result to regular relations:

Proposition 12 (Hennie 1965). *A one-tape deterministic TM running in $O(n)$ time is equivalent to a functional finite-state transducer.*

The Borodin-Trakhtenbrot Gap Theorem (Trakhtenbrot, 1964) states that expanded resources do not always expand the set of computable functions. In other words, it is possible that $O(n)$ is unnecessarily tight time bound for finite-state equivalence. A less tight time bound is now expressed with the little-o notation: $t(n) \in o(f(n))$ means that the upper bound $f(n)$ grows much faster than the running time $t(n)$ when n tends to infinity: $\lim_{n \rightarrow \infty} t(n)/f(n) = 0$.

Hartmanis (1968) and Trakhtenbrot (1964) showed independently that the time resource of a finite-state equivalent deterministic one-tape TM can be expanded from $O(n)$ to $o(n \log n)$ without expanding the characterized languages. More recently, Tadaki et al. (2010) showed that the bound $o(n \log n)$ applies also to nondeterministic one-tape TMs that explore all accepting computations.

Proposition 13 (Tadaki et al. 2010). *A one-tape TM running in $o(n \log n)$ time is equivalent to a finite automaton/transducer.*

A sufficient condition for finite-state equivalence of a TM is satisfied if the running time of the machine is bounded by a function $t(n)$ that is in $o(n \log n)$. For any reasonable function $t(n)$, this sufficient condition is decidable (Gajser, 2015). However, to decide finite-state equivalence of any TM, it would be necessary to consider all functions $t(n) \in o(n \log n)$.

We will assume a one-tape TM implementation for finite-fertility SCGs. The tape is initialized in such a way that f empty squares are reserved for latent cohorts at every cohort boundary.

We assume the representation of the grammar rules and the related application strategy by a functional transducer such as in Figure 1. Its optimization via the inward deterministic *bimachine* constructions (Yli-Jyrä, 2011; Hulden, 2011) optimizes the tape moves between derivation steps.

The parallel testing of all context conditions involves (i) the initialization step and (ii) a number of maintenance steps. The initialization step computes the validity of all context conditions at every tape squares in amortised $O(n)$ time. After this, the total amortised time needed to maintain the contexts is then bounded by the total number of moves needed to perform the subsequent rule applications.

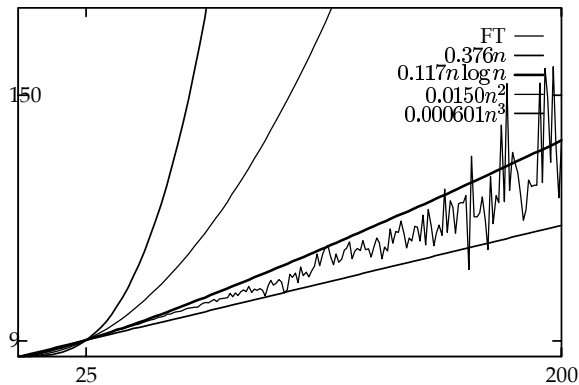


Figure 3: Average running time of CG-2 in Financial Times (according to Tapanainen 1999) seems to follow the curve $O(n \log n)$

Proposition 14. *The time used to maintain the context conditions is dominated by the time used to move between target cohorts.*

NM-SCGs based on a one-tape TM have now a regularity condition:

Proposition 15. *An NM-SCG is equivalent to a finite automaton/transducer if its one-tape TM implementation runs in $o(n \log n)$ time.*

This proposition can be compared to an interesting empirical observation by Tapanainen (1999) who reports experiments with a practical SCG (CG-2) system. According to the experiments, the average running time of the system follows closely the $O(n \log n)$ curve (Fig. 3).

On the basis of the experiments by Tapanainen, we cannot exclude the hypothesis that the asymptotic running time is actually in $o(n \log n)$. Whether the used grammar is actually equivalent to a finite-state transducer is not known.

If the given NM-SCG instance would be equivalent to a finite-state transducer, there would be a possibility to carry out monotonic SCG parsing in *linear time* and thus improve the parser's efficiency considerably. In case that the transducer is extremely large, the improvement remains solely as a theoretical possibility but the discovered regularity may still give valuable insight.

5.2 The $O(n)$ Time Bound

Hennie's finite-stateness condition (Hennie, 1965) for deterministic one-tape TMs and its generalization to nondeterministic one-tape TMs (Tadaki et al., 2010) are insightful and provide a method to construct the equivalent finite-state transducer when the finite-stateness condition is met.

LB	W	O	R	D	I	N	G	RB	...
q_0	q_1	q_2	$\boxed{q_3}$	q_4					
	q_7	$\boxed{q_6}$	q_5	\leftarrow					
	\hookrightarrow	q_8	$\boxed{q_9}$	q_{10}	q_{11}	...			

Figure 4: A crossing sequence between squares

A *Hennie machine* refers to a one-tape TM whose running time is $O(n)$. Hennie analysed the expressive power of such machines using the concept of *crossing sequence*, aka *schema* (Rabin, 1963; Trakhtenbrot, 1964). This concept is a powerful tool in the analysis of the behaviour of two-way automata and one-tape TMs.

A *crossing sequence* is the sequence of target states s_1, s_2, \dots visited by a TM when its pointer crosses the boundary between a pair of adjacent tape squares. States s_1, s_3, \dots are reached when the pointer moves forward and states s_2, s_4, \dots are reached when pointer moves backwards. Figure 4 shows how states are visited during a computation. The crossing sequence between the 3rd and the 4th squares is $(s_1, s_2, s_3) = (q_3, q_6, q_9)$.

Every Hennie machine satisfies the property that the length of its crossing sequences is bounded by an integer $k \in \mathbb{N}$. The finiteness of the crossing sequences of a given TM is undecidable (Průša, 2014) but if a finite upper bound k exists, this constant is computable (Kobayashi, 1985; Tadaki et al., 2010).

Finiteness of crossing sequences implies that the TM is equivalent to a finite-state automaton/transducer. Furthermore, the bound lets us construct this finite-state device. Unfortunately, the size complexity of the constructed machine is large in comparison to the original TM:

Proposition 16 (Průša 2014). *Each $|Q|$ -state, $|\Omega|$ -symbol deterministic Hennie machine can be simulated by a nondeterministic finite automaton with $2^{O(|\Omega| \log |Q|)}$ states.*

Testing the finite-stateness of already constructed TMs requires more effort than to design and construct machines that are immediately known to be Hennie machines. We will now mention a few immediate constructions.

Průša (2014)'s construction is based on a finite weight $w \in \mathbb{N}$ of the tape squares. Every time when a square is visited or passed, the weight associated with the square is reduced. Once the

weight is zero, further visits to the square are blocked.

Proposition 17 (Průša 2014). *A weight-reducing one-tape TM is a Hennie machine.*

Analogously, we can define an NM-SCG whose cohorts has a weight w that is reduced whenever the pointer of the associated TM implementation visits the corresponding square. The cohorts of such an NM-SCG have obviously a finite volume $v \leq w$ and can be changed at most w times.

Proposition 18. *A finite-fertility NM-SCG implemented by a weight-reducing one-tape TM is equivalent to a finite-state transducer.*

The second way to construct a Hennie-machine based NM-SCG is to set the maximum distance $m \in \mathbb{N} \cup \{\infty\}$ between adjacent rule applications.¹ When combined with the linear bound for rule applications, we obtain the $O(n)$ bound and finite-state equivalence:

Proposition 19. *A finite-fertility NM-SCG runs in $O(m(f+1)vn)$ time and is equivalent to a finite-state transducer if $m, f, v \in \mathbb{N}$.*

The third way is to assume fertility $f \in \mathbb{N}$ and $w = 1$. Since no square can be revisited, this forces the SCG to move constantly into one direction after all rule applications. This special case resembles the rewriting rules in finite-state phonology whose fundamental theorem (Johnson, 1972; Kaplan and Kay, 1994) states that if a phonological rule does not reapply to its own output (but instead moves on), it is regular.

The fourth way to construct a Hennie machine from an SCG is based on the number of times the context conditions for a cohort has to be updated. A monotonic SCG reduces the ambiguity of the sentence at every rule application. The reduced ambiguity causes occasional updates in context conditions of cohorts. Depending on the context conditions, such updates at a cohort boundary may have an infinite or finite bound. Due to functionality and inward determinism of the \Rightarrow -transducer, the pointer moves from one cohort to another only if the context conditions of the latter have changed as a result of a rule application. Thus, the number of context updates bound the number of moves:

Proposition 20. *If the context conditions can be updated only finitely often at every cohort, then the SCG is equivalent to a finite-state transducer.*

¹This approach was pursued and developed further by the current author in an earlier manuscript (Yli-Jyrä, unpublished) that is available on request.

6 Open Problems

6.1 Aperiodic Context Conditions

Yli-Jyrä (2003) showed that the context conditions used in a realistic Finite-State Intersection Grammar (FSIG) are not only regular but star-free. Since context conditions of SCG rules are strictly weaker than those of FSIG (Tapanainen, 1999), we have a strong conjecture that contexts in practical SCG are also star-free.

Star-free languages are definable in the monadic first-order logic of order, $FO[<]$, a decidable logic that is equivalent to LTL (Pnueli, 1977) and loop-free alternating finite automata (LF-AFA) (Salomaa and Yu, 2000). The states in an LF-AFA are totally ordered in such a way that every state is independent from all the preceding states in this order. This is a major restriction to the structure and expressive power of alternating finite automata.

While preserving possible star-freeness has led improvements in fundamental algorithms (Yli-Jyrä and Koskenniemi, 2004), we have not been able to solve the following open problem:

Open Problem 1. *Determine whether the construction of Hennie machines could benefit from star-freeness of the context conditions, possibly in combination with other conditions.*

6.2 Full Parsing

Reductionistic parsing (Koskenniemi, 1990; Maruyama, 1990; Voutilainen and Tapanainen, 1993; Gross, 1997; Eisner and Smith, 2005) is closely related to the consistency enforcing methods used in image recognition (Huffman, 1971; Clowes, 1971) and to the satisfiability in logic (Listenmaa, 2016). All these methods use some idea of domains that are then constrained.

Karlsson (1990) introduced the term *cohort* for ambiguity domains or lists of readings associated with tokens. Lauri Karttunen has then proposed (p.c., see also Voutilainen 1994) that the cohorts can be treated as strings and processed by finite-state transducers. This idea has been implemented later by others (Peltonen, 2011; Hulden, 2011).

Interestingly, the idea of processing ambiguity domains, i.e. cohorts, as strings is actually older than the SCG tradition. In the context of formal language theory, it dates back to Greibach (1973) and has been appreciated recently, e.g. by Okhotin (2013). What is interesting in Greibach's original use of cohorts is that these cohorts are used

to represent parse trees instead of just morphological ambiguity. The decomposition of trees and digraphs into local trees in the lexicon is actually due to the tradition of Categorical Grammar (Ajdukiewicz, 1935; Bar-Hillel, 1953; Lambek, 1958). This suggests an avenue for future SCG-related research.

Open Problem 2. *Develop an SCG grammar that performs full parsing on the basis of the structural ambiguity encoded into lexical categories.*

7 Conclusions

In this paper, the author has laid foundations for the analysis of the generative power of SCGs.

- The parsing is viewed as a derivation that resembles that of Generative Phonology.
- The equivalence between Nonmonotonic SCG and Turing machines is established, thus linking Constraint Grammar to Undecidability and the Chomsky hierarchy.
- Finite-fertility SCGs are shown to be context sensitive and running in quadratic time.
- A loose time bound $o(n \log n)$ for finite-state equivalent SCG instances (running on a TM) is provided and related to prior experiments.
- Specific conditions for constructing finite-state equivalent SCGs are given.
- Two open problems related to the potential of the star-freeness restriction of context conditions and the structural categories in the lexicon are presented.

The current work has demonstrated that the SCG formalism is not just a programming language for text linguistics but a formal framework that lends itself to connections to the richness of formal language theory and rigorous formal analysis of the related parsing complexities, culminating to attempts to reduce grammars into finite transducers.

Acknowledgements

The author has received funding as Research Fellow from the Academy of Finland (dec. No 270354 - A Usable Finite-State Model for Adequate Syntactic Complexity) and Clare Hall Fellow from the University of Helsinki (dec. RP 137/2013). The distance-based restriction of SCG has been studied by the author (Yli-Jyrä, unpublished) under earlier funding from the first agency (dec. 128536).

References

- Kazimierz Ajdukiewicz. 1935. Die syntaktische konnexität. In Storrs McCall, editor, *Polish Logic 1920-1939*, page 207231. Oxford University Press, Oxford. Translated from *Studia Philosophica*, 1, 1-27.
- Yehoshua Bar-Hillel. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:4758.
- J. R. Büchi. 1960. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6:66–92.
- Jean-Pierre Chanod and Pasi Tapanainen. 1995. A lexical interface for finite-state syntax. MLTT technical report, Rank Xerox Research Centre, Grenoble Laboratory, Grenoble, France, February 9.
- Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. Harper & Row, New York.
- Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, Massachusetts.
- M. B. Clowes. 1971. On seeing things. *Artificial Intelligence*, 2:79–116.
- Tino Didriksen, 2017. *Constraint Grammar Manual: 3rd version of the CG formalism variant*. GrammarSoft ApS, Denmark.
- Jason Eisner and Noah A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 30–41, Vancouver, British Columbia, October. Association for Computational Linguistics.
- Calvin C. Elgot. 1961. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51.
- David Gajser. 2015. *Verifying Time Complexity of Turing Machines*. Ph.D. thesis, University of Ljubljana, Department of Mathematics, Ljubljana, Slovenia.
- Sheila Greibach. 1973. The hardest context-free language. *SIAM Journal on Computing*, 2(4):304–310.
- Maurice Gross. 1997. The construction of local grammars. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*, chapter 11, pages 329–354. A Bradford Book, the MIT Press, Cambridge, MA, USA.
- Juri Hartmanis. 1968. Computational complexity of one-tape Turing machine computations. *J. ACM*, 15(2):325–339, April.
- Frederick C. Hennie. 1965. One-tape, off-line Turing machine computations. *Information and Control*, 8(6):553–578.
- D. A. Huffman. 1971. Impossible objects as nonsense sentences. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 6, pages 295–323. Edinburgh University Press, Edinburgh, Scotland.
- Måns Hulden. 2009. *Finite-State Machine Construction Methods and Algorithms for Phonology and Morphology*. Ph.D. thesis, Department of Linguistics, The University of Arizona.
- Mans Hulden. 2011. Constraint Grammar parsing with left and right sequential finite transducers. In *Proceedings of the 9th International Workshop on Finite State Methods and Natural Language Processing (FSMNLP 2011)*, pages 39–47, Blois, France, July. Association for Computational Linguistics.
- C. Douglas Johnson. 1972. *Formal Aspects of Phonological Description*. Number 3 in Monographs on linguistic analysis. Mouton, The Hague.
- Ronald M. Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378, September.
- Fred Karlsson. 1990. Constraint Grammar as a framework for parsing unrestricted text. In H. Karlgren, editor, *Proceedings of the 13th International Conference of Computational Linguistics*, volume 3, pages 168–173, Helsinki.
- Lauri Karttunen. 1994. Constructing lexical transducers. In *15th COLING 1994, Proceedings of the Conference*, volume 1, pages 406–411, Kyoto, Japan.
- K. Kobayashi. 1985. On the structure of one-tape non-deterministic Turing machine time hierarchy. *Theoretical Computer Science*, 40(2–3):175–193.
- Kimmo Koskenniemi. 1990. Finite-state parsing and disambiguation. In Hans Karlgren, editor, *13th COLING 1990, Proceedings of the Conference*, volume 2, pages 229–232, Helsinki, Finland, August.
- Sige-Yuki Kuroda. 1964. Classes of languages and linear-bounded automata. *Information and Control*, 7(2):207–223.
- Torbjörn Lager and Joakim Nivre. 2001. Part of speech tagging from a logical point of view. In P. de Groote, G. Morrill, and C. Retoré, editors, *Logical Aspects of Computational Linguistics*, volume 2099 of *Lecture Notes in Artificial Intelligence*, pages 212–227. Springer-Verlag.
- Joachim Lambek. 1958. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170.
- Nikolaj Lindberg and Martin Eineborg. 1998. Learning Constraint Grammar-style disambiguation rules using Inductive Logic Programming. In *36th ACL 1998, 17th COLING 1998, Proceedings of the Conference*, Montréal, Quebec, Canada, August 10-14.
- Inari Listenmaa. 2016. *Analysing Constraint Grammar with SAT*. Licentiate thesis, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden.

- Hiroshi Maruyama. 1990. Structural disambiguation with constraint propagation. In *28th ACL 1989, Proceedings of the Conference*, pages 31–38, Pittsburgh, Pennsylvania, June 6–9.
- John Myhill. 1960. Linear bounded automata. Wadd technical note, Wright Patterson AFB, Wright Air Development Division, Ohio, June.
- Alexander Okhotin. 2013. Inverse homomorphic characterizations of Conjunctive and Boolean Grammars. Technical Report 1080, Turku Centre for Computer Science, Turku.
- Janne Peltonen. 2011. Finite state Constraint Grammar parser. In *Proceedings of the NODALIDA 2011 workshop Constraint Grammar Applications, May 11, 2011*, volume 14 of *NEALT Proceedings Series*, Riga, Latvia.
- P. S. Peters and R. W. Ritchie. 1973. On the generative power of transformational grammars. *Information Sciences*, 6:49–83.
- Amir Pnueli. 1977. The temporal logic of programs. In *Proceedings of the IEEE 18th Annual Symposium on Foundations Computer Science*, pages 46–57, New York.
- Daniel Průša. 2014. Weight-reducing Hennie machines and their descriptive complexity. In *Language and Automata Theory and Applications: 8th International Conference, LATA 2014, Madrid, Spain, March 10–14, 2014. Proceedings*, pages 553–564, Cham. Springer International Publishing.
- Michael O. Rabin. 1963. Real time computation. *Israel Journal of Mathematics*, 1(4):203–211.
- Eric Sven Ristad. 1990. Computational structure of generative phonology and its relation to language comprehension. In *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics, ACL '90*, pages 235–242, Pittsburgh, Pennsylvania.
- Kai Salomaa and Sheng Yu. 2000. Alternating finite automata and star-free languages. *Theoretical Computer Science*, 234:167–176.
- Wojciech Skut, Stefan Ulrich, and Kathrine Hammer-vold. 2004. A bimachine compiler for ranked tagging rules. In *Proc. 20th Int'l Conf. on Computational Linguistics, COLING '04*, Stroudsburg, PA, USA.
- Kohtaro Tadaki, Tomoyuki Yamakami, and Jack C. H. Lin. 2010. Theory of one-tape linear-time Turing machines. *Theoretical Computer Science*, 411(1):22–43.
- Pasi Tapanainen. 1996. *The Constraint Grammar Parser CG-2*, volume 27 of *Publications*. Department of General Linguistics, University of Helsinki.
- Pasi Tapanainen. 1999. *Parsing in two frameworks: finite-state and functional dependency grammar*. Ph.D. thesis, University of Helsinki, Finland, 1 December.
- B. A. Trakhtenbrot. 1961. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 140:326–329. In Russian.
- Boris A. Trakhtenbrot. 1964. Turing computations with logarithmic delay (in Russian). *Algebra i Logika*, pages 33–34. English translation in U. of California Computing Center, Tech. Report. No. 5, Berkeley, CA, 1966.
- Atro Voutilainen and Pasi Tapanainen. 1993. Ambiguity resolution in a reductionistic parser. In *6th EACL 1993, Proceedings of the Conference*, pages 394–403, Utrecht, The Netherlands.
- Atro Voutilainen. 1994. *Designing a Parsing Grammar*. Number 22 in Publications of the Department of General Linguistics, University of Helsinki. Yliopistopaino, Helsinki.
- Anssi Mikael Yli-Jyrä and Kimmo Koskenniemi. 2004. Compiling contextual restrictions on strings into finite-state automata. In Loek Cleophas and Bruce W. Watson, editors, *The Eindhoven FASTAR Days, Proceedings*, number 04/40 in Computer Science Reports, Eindhoven, The Netherlands, December. Technische Universiteit Eindhoven.
- Anssi Mikael Yli-Jyrä. 2003. Describing syntax with star-free regular expressions. In *11th EACL 2003, Proceedings of the Conference*, pages 379–386, Agro Hotel, Budapest, Hungary, April 12–17.
- Anssi Yli-Jyrä. 2008. Transducers from parallel replace rules and modes with generalized lenient composition. In *Finite-State Methods and Natural Language Processing, 6th International Workshop, FSMNL-2007*, pages 197–212, Potsdam. Potsdam University Press.
- Anssi Yli-Jyrä. 2011. An efficient constraint grammar parser based on inward deterministic automata. In *Proceedings of the NODALIDA 2011 workshop Constraint Grammar Applications, May 11, 2011*, volume 14 of *NEALT Proceedings Series*, Riga, Latvia.
- Anssi Yli-Jyrä. unpublished. Efficient context-sensitive rewriting with inward deterministic transducers. Manuscript, 11 pages. Archived to EasyChair as a submission to PSC 2010 (Prague Stringology Conference 2010).