Describing Syntax with Star-Free Regular Expressions

Anssi Yli-Jyrä
Department of General Linguistics, P.O. Box 9, FIN-00014 Univ. of Helsinki, Finland
anssi.yli-jyra@helsinki.fi

Abstract

Syntactic constraints in Koskenniemi's Finite-State Intersection Grammar (FSIG) are logically less complex than their formalism (Koskenniemi et al., 1992) would suggest: It turns out that although the constraints in Voutilainen's (1994) FSIG description of English make use of several extensions to regular expressions, the description as a whole reduces to a finite combination of union, complement and concatenation. This is an essential improvement to the *descriptive complexity* of ENGFSIG. The result opens a door for further analysis of logical properties and possible optimizations in the FSIG descriptions. The proof contains a new formula for compiling Koskenniemi's restriction operation without any marker symbols.

1 Introduction

For many years, various finite-state models of language (Roche and Schabes, 1997) have been used in surface-syntactic parsing. These models can process *local* syntactic ambiguity efficiently. However, because the formalism of Finite-State Intersection Grammar (Koskenniemi, 1990; Koskenniemi et al., 1992) allows full regular expressions, its parsing is sometimes inefficient (Tapanainen, 1997); many FSIG constraint automata can reduce ambiguity only after they have scanned the *whole sentence*.

Regular expressions in FSIG can be viewed as a grammar-writing tool that should be as flexible as possible. This viewpoint has led to introduction of new features into the formalism (Koskenniemi et al., 1992). It is, however, very difficult to make any a priori generalizations of the structural properties of automata as long as we allow unrestricted use of regular expressions.

A complementary view is to analyze the properties of languages described by FSIG regular expressions. We can carry out the analysis by checking whether the languages can be described with a restricted class of regular expressions. For many such classes of expressions, there also exists a group-theoretic characterization (Pin, 1986). Moreover, if the analyzed regular language has favorable properties, some problems, e.g. the string membership problem, can be solved faster by means of specialized algorithms.

A language can be described with a *star-free* regular expression if it can be constructed from alphabet symbols by application of union ($A \cup B$), complementation ($\sim A$) and finite concatenation (AB), that is, without the Kleene closure (A^*). The theoretical importance of this class of languages is supported by its characterization in terms of finite aperiodic syntactic monoids (Schützenberger, 1965) and by its definability in first-order logic over strings (McNaughton and Papert, 1971). The class has also a lot of practical importance, because many languages in it admit extremely simple implementations (ibid.).

The question of the star-freeness restriction on FSIG constraints has not been studied before, possibly because of the following observations:

(i) An acyclic automaton representing readings of the sentence has a central role in FSIG parsing (Tapanainen, 1997). Star-freeness of the constraints is a minor restriction when compared to the finiteness of this language.

(ii) If automata states are encoded as "traces" into strings, any regular language can be represented as a homomorphic image of a (local) star-free language (Medvedev, 1964). Such an encoding is possible in a two-level view of the FSIG framework (Koskenniemi, 1997), where the morphological reading of the sentence is a homomorphic image of a level representing syntactically annotated readings.

(iii) Given a finite automaton or a regular expression, checking star-freeness of the described language is an intractable (see 2.2) problem.

(iv) Automatical methods to derive star-free regular expressions from another representations procuce long and unintuitive expressions (Matz et al., 1995).

From my point of view, these observations miss some important perspectives: Firstly (i), it is important to understand that a finite-state intersection grammar is also a description of a language with a structure of its own, independent of the acyclic sentence automaton. Secondly (ii), a realistic FSIG description is linguistically motivated and leaves little room for encoding of traces that could technically make the grammar star-free. Thirdly (iii), heuristic methods can be used to solve many large star-freeness problems in practice. Fourthly (iv), it is often possible to find star-free regular expressions that are short and illustrative, as it turns out in this paper.

Any automaton recognizing a non-star-free language has a factor that induces a nontrivial permutation of the state space. For example, the parity language $0^*(10^*10^*)^*$ contains strings with an even number of occurrences of the factor "1". Intuitively, it seems improbable that similar counting constraints occur in natural language grammars. However, many regular expressions in Voutilainen's ENGFSIG (1994) involve the Kleene star. If we can explain why this does not affect the star-freeness of the language, we probably know more about the grammar itself.

A significant contribution of this paper is the human-readable construction that rephrases ENGFSIG (Voutilainen, 1994) constraints without the Kleene star. To make the construction more systematic I first outline the framework of FSIG and define its star-freeness problem. After this I explore stars in the ENGFSIG description and reduce regular expressions in the description into their star-free equivalents. This approach extends to a closure property of the star-free regular languages under the restriction operator (of FSIG).

2 Finite-State Intersection Grammar

In this section I define a class of finite-state intersection grammars and explain the star-freeness problem specific to them. The FSIG framework developed here is based on the work of Koskenniemi, Tapanainen and Voutilainen (1992).

2.1 Definitions

I start by making my terminology on the strings described precise. In FSIG, a sentence is seen as a *syntactically annotated string* that is exemplified in the following string:

```
"@@ time   N NOM   SG   @SUBJ @
    fly    V PRES  SG3  @MV   @
    like   PREP         @ADVL @
    an     DET ART SG   @>N   @
    arrow  N NOM   SG   @P<<  @@"
```

This string of *tags* represents a possible syntactic structure for the sentence 'time flies like an arrow'. In the example, all the tags that start with an @-sign contribute to the syntactic analysis. In this example, the tags `@@` and `@` denote sentence and word boundaries, respectively. They delimit *word analyses*. For each word, the *morphological analysis* like "`time N NOM SG`" precedes the tags that denote the *syntactic function* of the word. Syntactic tags specify, in this example, that the word 'time' functions as the subject (`@SUBJ`), and the word 'arrow' is the complement for a preposition on the left (`@P<<`).

An *(unweighted) finite-state intersection grammar* is a tuple $G = (\Sigma_B, \Sigma_W, \Sigma_F, B, W, F, C, d)$, in which

- $\Sigma_B, \Sigma_W, \Sigma_F \subset \Sigma$ are disjoint alphabets,
- $B \subseteq \Sigma_B$ is a set of *delimiters* that can appear before and after word analyses,
- $W \subseteq \Sigma_W^+$ is a finite lexicon of morphological analyses,
- $F \subseteq \Sigma_F^+$ is a finite set of tag strings that denote syntactic functions, and,

- $C = \{c_1^d, c_2^d, \ldots, c_n^d\}$ is a set of finite-state constraints (regular languages) with the alphabet Σ, where
- $d \in \mathbb{N}$ is a finite bound for the maximum center-embedding depth in the constraints.

The regular set $D = B(WFB)^+$ is the *domain of annotated strings*. The language described by the grammar G is defined by the set $L(G) = D \cap c_1^d \cap c_2^d \cdots \cap c_k^d, c_{k+1}^d \cdots \cap c_n^d$. The first k constraints apply locally to each word, matching morphological analyses with potential syntactic functions. I call them *local lexical constraints*. All the constraints are expressed by means of *FSIG regular expressions*.

Any symbol $a \in \Sigma$, as well as any symbol set $\{a_1, a_2, \ldots, a_m\}$, $a_1, a_2, \ldots, a_m \in \Sigma$, are valid FSIG regular expressions. The language consisting of the empty string is denoted with ϵ (or $[\,]$ in the FSIG notation). In addition to the simple operators (Table 1) that combine expressions A and B, FSIG regular expressions make use of the *restriction* operator (Koskenniemi, 1983). It has the following syntax:

$$X \Rightarrow \ LC_1 \ \_ \ RC_1, \ \ LC_2 \ \_ \ RC_2, \ \ \ldots, LC_n \ \_ \ RC_n$$

The operands $X, LC_1, \ldots, RC_n$ are FSIG regular expressions. The semantics of the whole expression is as follows: Whenever a substring $x \in X$ occurs in the string w, its context must match at least one of the patterns $LC_i \ \_ \ RC_i$, $i = 1..n$. When there are overlapping occurrences of the *center* X, the string w is rejected if any of the occurrences infringes the restriction (this is the *strict interpretation* of the operator).

A *center-embedded clause* is an embedded clause that is not the leftmost neither the rightmost constituent in its matrix clause. In the ENGFSIG representation, a finite center-embedded clause is separated from its matrix clause with a pair of delimiters @<$\in B$ and @>$\in B$. Sequential clause boundaries are denoted (ambiguously) with the delimiter @/$\in B$. Special constants (Koskenniemi et al., 1992) are used to facilitate description of complex patterns involving the delimiter symbols Σ_B, $\Sigma_B \supseteq B, B = \{$@, @<, @>, @/, @@$\}$. The intuitive meaning of the constants in Table 2 is as follows: The *dot* ⊡ accepts tag sequences of Σ_W and Σ_F inside word analyses, the expression ⊡ accepts tag sequences of Σ_W, Σ_F and @, and the constant @<> accepts a center-embedded clause with possible nested center-embeddings. The *dot-dot* ⊡ differs from the expression ⊡ by accepting anything within the same clause, including center-embedded clauses. Finally, the *dots* ⊡ accepts anything at the same level of center-embedding.

The FSIG notation	The current notation	Precedence	Semantics of the expression
$[A\,]$	$[A]$	(6)	A
$(A\,)$	$A \cup \epsilon$	(6)	$A \cup \epsilon$
$-A$	$\overline{A}$ or $\sim A$	5	$\{x \mid x \in \Sigma^* \wedge x \notin A\}$
$A+$	A^+	4	AA^*
$A*$	A^*	4	$\cup_{i=0}^{\infty} \overbrace{AA...A}^{i}$
$\backslash A$	$\overline{\$A}$ or $\sim \$A$	3	$\sim[\Sigma^* A \Sigma^*]$
AB	AB	2	$\{xy \mid x \in A \wedge y \in B\}$
$A \mid B$	$A \cup B$	1	$\{x \mid x \in A \vee x \in B\}$
$A \ \& \ B$	$A \cap B$	1	$\sim[\sim A \cup \sim B]$
N/A	$A - B$	1	$A \cap \sim B$

Table 1: Combinations of expressions A and B.

FSIG	Current	Semantics
.	⊡	$\sim\$\,\Sigma_B$
>..<	⊡	$[$⊡ $\cup \{$@$\}]^*$
@<>	@<>	(explained in the text)
..	⊡	$[$⊡ $\cup \{$@$\} \cup$ @<>$]^*$
...	⊡	$[$⊡ $\cup \{$@, @/$\} \cup$ @<>$]^*$

Table 2: The special constant expressions.

The *parameter* d specifying the maximum depth of center-embedding is an essential element of the FSIG regular expressions. The bound is needed to compile constraints that contain the constant @<>, because the idealized language described by the constant @<> is context-free, in fact, a *counter language* in terms of Schützenberger (1962). In a practical implementation (Koskenniemi et al., 1992), the language @<> is approximated with a regular language. I denote the approximation using the parameterized expression @<>^{d} (Figure 1). The generic expressions @<>^{i}, $i \in 1, 2, 3, \ldots$, as well as the constants ⊡^{d} and ⊡^{d} are defined as follows:

$$\begin{aligned}
\text{@<>}^{\,0} &= \epsilon \\
\text{@<>}^{\,i} &= \text{@<}\,[\,[\,\boxdot \cup \{\text{@, @/}\}]^*\ \text{@<>}^{\,i-1}\,]^* \\
&\qquad [\,\boxdot \cup \{\text{@, @/}\}]^*\ \text{@>} \\
\boxed{\cdot\cdot}^{\,d} &= [\boxdot \cup \{\text{@}\} \cup \text{@<>}^d]^* \\
\boxed{\cdots}^{\,d} &= [\boxdot \cup \{\text{@, @/}\} \cup \text{@<>}^d]^*
\end{aligned}$$

Finally, FSIG regular expressions may contain user-defined macros as subexpressions. They can have a constant value or take other expressions as arguments.

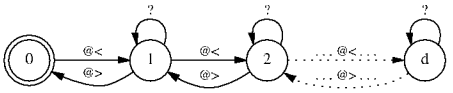

Figure 1: A finite automaton ($? = \Sigma - \{$@<, @>$\}$) that visualizes the semantics of @<>^{d}.

2.2 Star-freeness problem for an FSIG

The problem I want to solve for an FSIG is the star-freeness problem. It is, given a grammar G, to determine whether the language $L(G)$ is *star-free* i.e. whether it can be constructed from alphabet symbols by application of the boolean operators ($\cup, \sim$) and concatenation.

Proposition 1. For a regular language L, the following properties are equivalent:

- the language L is *star-free*,
- there is a *starfree* regular expression, based on concatenation and the boolean operators, that describes the language L,
- the *syntactic monoid* (McNaughton and Papert, 1968) that is canonically assigned to the language L is *aperiodic* (Schützenberger, 1965),
- the language L is definable in propositional *linear temporal logic* (Kamp, 1968), and,
- the language L is definable in a *first-order logic that is interpreted over finite strings* (McNaughton and Papert, 1971).

Sometimes star-freeness of a language can be shown by means of closure properties of star-free languages. To start with, finite regular languages are star-free (especially $\emptyset$, ϵ, a, and Γ, where $\emptyset$ denotes the empty set of strings, $a \in \Sigma$, and $\Gamma \subseteq \Sigma$) The Kleene closure of any subset $\Gamma \subseteq \Sigma$ is also star-free, because $\Gamma^* = \overline{\overline{\emptyset}[\Sigma - \Gamma]\overline{\emptyset}}$. If A and B are star-free languages, then we know that at least the following languages are star-free (McNaughton and Papert, 1971):

$$AB \quad \overline{A} \quad \overline{\$A} \quad A \cup B \quad A \cap B \quad A - B$$

It is also possible that the language of a regular expression is star-free although the expression contains the Kleene star operator. Therefore, the method based on the properties of the syntactic monoid of the language is important. The syntactic monoid is usually difficult to compute manually, and some programs, e.g. AMoRe (Matz et al., 1995) are designed to facilitate these computations and aperiodicity testing. The *aperiodicity problem* is, however, computationally intractable (PSPACE-complete) both for regular expressions (Bernátsky, 1997) and for deterministic automata (Cho and Huynh, 1991).

It is often possible to heuristically prove the star-freeness property by inventing an equivalent star-free expression.

Proposition 2. In order to show that a finite-state intersection grammar G is star-free, it is sufficient to show that:

- the domain $B(WFB)^+$ is star-free,
- the local lexical constraints $c_1^d, c_2^d, \ldots, c_k^d$ are star-free,
- the constants $\boxdot$, $\boxed{\cdot\cdot}$, $\boxed{\cdots}$, $\boxed{>\cdots<}$, @<> and other subexpressions in the constraints are star-free, and,
- the star-free languages are closed under the operators that combine the subexpressions into the constraints $c_{k+1}^d, c_{k+2}^d, \ldots, c_n^d$.

3 The reduction of ENGFSIG into star-free expressions

3.1 The domain of annotated strings

Because the alphabets Σ_B, Σ_W and Σ_F are disjoint and the sets B, W and F do not contain an empty string, the set $S = \Sigma_B^+(\Sigma_W^+\Sigma_F^+\Sigma_B^+)^+$ can be expressed as $[\Sigma_B^+\Sigma_W\Sigma^*] \cap [\Sigma^*\Sigma_F\Sigma_B^+] \cap \sim\$[\Sigma_B[\Sigma - \Sigma_B - \Sigma_W]] \cap \sim\$[\Sigma_W[\Sigma - \Sigma_W - \Sigma_F]] \cap \sim\$[\Sigma_F[\Sigma - \Sigma_F - \Sigma_B]]$. The remaining question is, whether the sets B, W and F are star-free languages. In the case of ENGFSIG they are finite, and therefore, each of them is expressible with a star-free regular expression. Hence the iteration in $B(WFB)^+$ translates to $S \cap [B\Sigma_W\Sigma^*] \cap [\Sigma^*\Sigma_F B] \cap \sim\$[\Sigma_B[\Sigma_W^* - W]\Sigma_F] \cap \sim\$[\Sigma_W[\Sigma_F^* - F]\Sigma_B] \cap \sim\$[\Sigma_F[\Sigma_B^* - B]\Sigma_W]$

3.2 The local lexical constraints

The relation between the morphological analyses and the allowed syntactic functions can be implemented either with one or two levels (Koskenniemi, 1997) in a practical FSIG parser. In the grammar G, this relation corresponds to a set of lexical constraints $c_1^d, c_2^d, \ldots, c_k^d$.

In the case of ENGFSIG, the local lexical constraints reduce to a boolean combination of languages of the form $\$t$, $t \in \Sigma_W \cup \Sigma_F$, because the tag positions in the strings of W and F are fixed by a convention that partly reflects the simple morpheme structure of English words. Let the lexical constraints in conjunction with the domain D describe the set $B\,(L_{WF}B)^+$, $L_{WF} \subseteq WF$. The conformance to this property is enforced by the following star-free constraint:

$$D \cap \overline{\Sigma^*\Sigma_B\,[\Sigma_W^+\Sigma_F^+ - L_{WF}]\,\Sigma_B\Sigma^*}$$

3.3 The constant expressions

It is pretty easy to see that the expressions $\texttt{@<>}^0$ and $\texttt{@<>}^1$ are star-free. I managed to find an inductive derivation for general case $\texttt{@<>}^i, i \in 1, 2, 3, \ldots$. The following defines the dependent constants $\texttt{@<>}^i$, $\boxed{\cdots}^i$ and $\boxed{\cdot\cdot}^i$, as well as the constant $\boxed{>\cdots<}$ with star-free operators:

$$\begin{aligned}
\boxed{\cdots}^0 &= \overline{\$\{\texttt{@<}, \texttt{@>}, \texttt{@@}\}} \\
\texttt{@<>}^0 &= \epsilon \\
\texttt{@<}^i &= [\boxed{\cdots}^{i-1}\,\texttt{@<}] \ldots [\boxed{\cdots}^0\texttt{@<}] \\
\texttt{@>}^i &= [\boxed{\cdots}^0\,\texttt{@>}] \ldots [\boxed{\cdots}^{i-1}\texttt{@>}] \\
\boxed{\cdots}^i &= \overline{\$\texttt{@@}} \cap \overline{\$[\texttt{@<}\,\texttt{@<}^i]} \cap \overline{\$[\texttt{@>}\,\texttt{@>}^i]} \\
&\quad \cap \overline{\boxed{\cdots}^{i-1}\texttt{@>}\,\Sigma^* \cap \Sigma^*\,\texttt{@<}\,\boxed{\cdots}^{i-1}} \\
\texttt{@<>}^i &= \texttt{@<}\,\boxed{\cdots}^{i-1}\,\texttt{@>} \\
\boxed{\cdot\cdot}^i &= \boxed{\cdots}^i \cap \overline{\boxed{\cdots}^i\,\texttt{@/}\,\boxed{\cdots}^i} \\
\boxed{>\cdots<} &= \sim\$[\Sigma_B - \texttt{@}]
\end{aligned}$$

3.4 The subexpressions with the Kleene star

The version of ENGFSIG studied contains 983 subexpressions (of 221 types) containing the Kleene star operator. Each iterated subexpression seems to have two components: (i) a *domain of iteration* which specifies what kind of unit is iterated, and (ii) a *condition* which specifies the necessary property for each unit. By unifying every left-oriented domain of iteration (e.g. $\boxdot$ @) with the corresponding right-oriented domain (e.g. @ $\boxdot$), I identified four variants of domains (Table 3).

Domain	Freq.	Iterated unit (R)	Conditions
R/L_{word}	938	@ $\boxdot$	196
R/L_{clause}	42	@/ $\boxed{\cdot\cdot}$	22
R/L_{after}	2	{@/, @<} $\boxed{\cdot\cdot}$	2
R/L_{embedded}	1	@< $\boxed{\cdot\cdot}$	1

Table 3: The four domains of iteration.

The domain and the condition are seldom separated in a ENGFSIG regular expression. Instead, the condition is usually inside the Kleene closure that specifies the domain. For example in the subexpression [@ $\boxdot$ @>A $\boxdot$]*, the domain is a word preceded by a word boundary (@ $\boxdot$), and the condition is that each word must be an adjective-premodifier.

Iteration of the right-oriented domains corresponds to the following star-free regular expressions:

$$\begin{aligned}
R^*_{\text{word}} &= \epsilon \cup [\texttt{@}\ \ \boxed{>\cdots<}^d] \\
R^*_{\text{clause}} &= \epsilon \cup [\texttt{@/}\ \boxed{\cdots}^d] \\
R^*_{\text{after}} &= \epsilon \cup [\{\texttt{@/}, \texttt{@<}\}\ [\ \overline{\boxed{\cdots}^d\texttt{@>}\,\Sigma^*} \cap \\
&\qquad \overline{\$\texttt{@@}} \cap \overline{\$[\texttt{@>}\,\texttt{@>}^d]}\ ]] \\
R^*_{\text{embedded}} &= \epsilon \cup [\quad \texttt{@<}\ [\ \overline{\boxed{\cdots}^d\texttt{@>}\,\Sigma^*} \cap \\
&\qquad \overline{\Sigma^*\texttt{@/}\,\overline{\boxed{\cdots}^d\,\texttt{@>}\,\Sigma^*}} \cap \overline{\$\texttt{@@}} \cap \overline{\$[\texttt{@>}\,\texttt{@>}^d]} \cap \\
&\qquad \overline{\overline{\$\texttt{@<}\,\texttt{@/}\,\Sigma^*} \cap \overline{\Sigma^*\,\texttt{@/}\,\overline{\$\texttt{@>}}}}\ ]]
\end{aligned}$$

ENGFSIG associates typically very simple conditions with the domain of iteration. In the star-free form of a starred expression, the domain of iteration and the associated condition are defined separately and then combined under the intersection operator. In the following, I give some examples of possible conditions and how they are represented in separation from the domain:

- The phrase "every @>N 6,000 @>N miles N @ADVL" satisfies the constraint "N $\boxdot$ @ADVL $\Rightarrow$ every $\boxdot$ @>N @ [$\boxdot$ @>N $\boxdot$ @]* $\boxdot$ _ ". In [$\boxdot$ @>N $\boxdot$ @]*, the domain of iteration is L_{word}, a reverse counterpart to R_{word}. The corresponding condition is as follows: $\sim[\,\overline{\$\{\texttt{@}, \texttt{@>N}\}}\ \texttt{@}\ \Sigma^*] \cap \sim\$[\texttt{@}\ \overline{\$\ \texttt{@>N}}\ \texttt{@}]$.

- Conditions often specify the absence of a word (or a tag). The closure [[$\boxdot$ $\cap$ $\overline{\$\text{DET}}$] @>N[$\boxdot$ $\cap$ $\overline{\$\text{DET}}$] @]* can be simplified as follows: [$\boxdot$ @>N $\boxdot$ @]* $\cap$ $\overline{\$\text{DET}}$.

- If the domain of iteration is the clause R_{clause}, then the condition may require that each clause contains a main verb (@MV). Such a condition translates as follows: $\sim[\Sigma^*\ \texttt{@/}\ \overline{\$\{\texttt{@/}, \text{MV}\}}] \cap \sim\$[\texttt{@/}\ \overline{\$\text{MV}}\ \texttt{@/}]$.

- Sometimes the iterated clause R_{clause} is not allowed to contain center-embeddings. This condition reads: $\sim\$\{\texttt{@<}, \texttt{@>}\}$.

ENGFSIG contains only 12 examples of nested Kleene stars. One example is in the following:

[@/ [⊡ [@comma|CC] ⊡ @]* ⊡ @CC ⊡⋯]*

In all these cases, the inner application of Kleene star can be expressed as a condition applying to the domain of the outer iteration level.

3.5 The restriction operator

In Section 3.2, I have described how the local lexical constraints can be represented without the Kleene star operator. In addition to these, there are 2657 more complicated constraints. The schematic equivalences presented in Sections 3.3 – 3.4 can transform 1554 of these into a star-free form. However, there still remain 1103 constraints that use the restriction operator ($\Rightarrow$). To complete the proof of the star-freeness of ENGFSIG, I show that star-free languages are closed under the restriction operation (as in FSIG).

Compilation of the restriction operator (as in Two-Level Morphology) has been solved by means of marker symbols and transducers (Karttunen et al., 1987; Kaplan and Kay, 1994). To compile the restriction as in FSIG, Tapanainen (1992) used also a method that is perhaps most easily described with transducers. When there is only one context LC_1 _ RC_1, the restriction operator (as in TWOL and in FSIG) reduces to the following *star-free formula* (Karttunen et al., 1987):

$$\overline{\overline{\Sigma^* LC_1}\ X\ \overline{\emptyset}} \cap \overline{\overline{\emptyset}\ X\ \overline{RC_1 \Sigma^*}}$$

I generalize this special case in the following *new formula for n contexts* LC_i _ RC_i, $i = 1..n$:

$$\bigcap_{\mathcal{F}=\{\}}^{\mathcal{S}} \overline{[\bigcap_{i=1}^{n} \overline{\phi(i, \mathcal{S}-\mathcal{F})\, LC_i}\ ]\ X\ \bigcap_{i=1}^{n} \overline{RC_i\, \phi(i, \mathcal{F})}}$$

where $\mathcal{S} = \{1, 2, \ldots, n\}$ and

$$\phi(i, \mathcal{F}) = \begin{cases} \Sigma^* & \text{if } i \in \mathcal{F}; \\ \emptyset & \text{otherwise;} \end{cases}$$

The above formula does not use markers, transducers, nor the Kleene star. Intuitively, it says that the string is rejected on the basis of the match of X, if each of the n contexts around a match of X fails at least on one side ($\phi(i, \mathcal{S}-\mathcal{F})$ ... $\phi(i, \mathcal{F})$). There are 2^n different ways ($\mathcal{F} = \{\}$, $\{1\}$, $\{2\}$, $\{1,2\}$,..$\{1,2,\ldots,n\}$) to choose a failing side for every member in the set of contexts LC_i _ RC_i, $i = 1..n$.

4 Experiments

I initially extracted the starry subexpressions from the ENGFSIG grammar and classified them using a Perl script. At a later stage, I developed a regular expression preprocessor that automated many tasks. The results were compared across different formulas in order to find possible differences.

The preprocessor could output a script where operands for each restriction operator were defined (and compiled into automata) before the operator was applied. Every bunch of operand definitions was followed by a formula that implemented the restriction operator with a required number of contexts. In order to reduce the number of contexts, I gathered unilateral contexts with the preprocessor.

I developed and tested the presented equivalences using the Xerox Finite-State Tool (v.7.4.0). My new formula for the restriction operator produced automata that were equivalent to the output of Tapanainen's rule compiler (Koskenniemi et al., 1992), which was actually used during the development of ENGFSIG.

I also compared these automata to the ones that would result from using Kaplan and Kay's (1994) method and some variants of it. Some differences in the results suggest that they use another interpretation for the (compound) restriction operator. According to that interpretation, overlapping centers are not restricted conjunctively, sometimes resulting in a bigger language.

Simple optimizations in the formula for an n-context restriction made a notable difference in compilation time. When I compiled a 7-context restriction (this was a striking exception in ENGFSIG), an unoptimized version of my formula was very slow (9 min.) compared to a transducer-based method (34.8 sec.), while an optimized version was roughly as efficient (35.5 sec.). In this example, the number of (outer) conjuncts in my formula was quite high (2^7). The new formula is at its best in the typical case when the number of contexts is smaller than seven.

I did not make experiments with starry subexpressions because they are relatively small and fast to compile anyway.

5 Discussion

The schematic equivalences presented suggest alternative ways to compile some special cases of Kleene star. The compilation of Kleene closures into deterministic automata involves determinization that is based on the subset construction. On the basis of the equivalences presented here it may be possible to identify more cases for which we can find specialized determinization algorithms (Mohri, 1995).

The new formula for the restriction operator has one extra advantage over compilation methods that are based on marker symbols and transducers (Kaplan and Kay, 1994). In these methods, the markers have to be eliminated from the final language. Usually this requires determinization using the costly subset construction. The new formula does not involve markers and it therefore only needs to apply determinization at smaller sub-formulas.

Methods that reduce the size of constraint automata can contribute to an efficient solution for the FSIG parsing problem (Koskenniemi, 1997) by producing a smaller representation for the grammar. Tapanainen (1992) has developed special optimizations that apply to automata during their construction. The current paper suggests manipulation of FSIG regular expressions before they are compiled into deterministic automata. The value of this approach is based on the fact that the construction of a deterministic automaton from a regular expression is, in the worst-case, exponential.

The current paper provides the FSIG framework with a grammar semantics that is completely based on regular languages and a one-level representation. Our new formula for an n-context restriction operator does not make use of transducers (Tapanainen, 1992) nor markers. In the absence of such complications, axioms for regular expressions (Antimirov and Mosses, 1994) become much more usable and may lead to essential simplifications in the individual constraints (see Section 4) and in the grammar altogether.

The new formula for the restriction operator enables us to split an n-context restriction into 2^n separate constraints (under intersection), each of which can be simplified, compiled and applied separately. It is also possible to compile the FSIG regular expressions directly into a single *alternating finite automaton* where intersection and complementation can occur inside the grammar automaton. Manipulation of alternating automata (Vardi, 1995) may help us to avoid the state explosion that is the main problem with deterministic automata in FSIG parsing (Tapanainen, 1997).

Finally, the main contribution of this paper is to show that ENGFSIG describes a star-free set of strings. It seems probable that this narrowing could be added to the FSIG framework in general.

The *computational complexity* of many important decision problems for the FSIG grammars remains intractable in spite of the star-freeness property (Sistla and Clarke, 1985). Nevertheless, the improved descriptive complexity allows us to simplify some algorithms; we can, for example, implement the grammar with the class of *loop-free* alternating automata (Salomaa and Yu, 2000). Moreover, the restriction also means that the grammar is definable in a *first-order logic* that is interpreted over finite strings (McNaughton and Papert, 1971). This simplification is relevant to reconstruction of FSIG and similar finite-state models with logical specifications (Vaillette, 2001; Lager and Nivre, 2001).

6 Conclusion

In this paper, the ENGFSIG description as a whole is shown to be a regular expression that reduces to a combination of union, complementation and finite concatenation. The current work has theoretical and practical consequences in processing of ENGFSIG (or similar) descriptions, context restrictions in the Two-Level Morphology, and Kleene closures in wider domains.

Acknowledgments

This work was supported by NorFA Ph.D. programme. I am grateful to Atro Voutilainen (and Connexor) for putting to my disposal the ENGFSIG description. I would also like to thank especially Lauri Carlson, as well as Voutilainen, Kimmo Koskenniemi, and the referees for useful comments on this paper.

References

Valentin M. Antimirov and Peter D. Mosses. 1994. Rewriting extended regular expressions. In G. Rozenberg and A. Salomaa, editors, *Developments in Language Theory — at the Crossroads of Mathematics, Computer Science and Biology*, pages 195–209. World Scientific.

László Bernátsky. 1997. Regular expression star-freeness is PSPACE-complete. *Acta Cybernetica*, 13(1):1–21.

Sang Cho and Dung T. Huynh. 1991. Finite-automaton aperiodicity is PSPACE-complete. *Theoretical Computer Science*, 88:99–116.

Johan A.W. Kamp. 1968. *Tense Logic and the Theory of Linear Order*. Ph.D. thesis, Univ. of California, Los Angeles.

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

Lauri Karttunen, Kimmo Koskenniemi, and Ronald M. Kaplan. 1987. A compiler for two-level phonological rules. Technical Report CSLI-87-108, CSLI, Stanford University.

Kimmo Koskenniemi, Pasi Tapanainen, and Atro Voutilainen. 1992. Compiling and using finite-state syntactic rules. In *Proc. COLING'92*, volume I, pages 156–162. Nantes, France.

Kimmo Koskenniemi. 1983. *Two-level morphology: a general computational model for word-form recognition and production*. Nr. 11 in Publications of the Dept. of General Linguistics. University of Helsinki.

Kimmo Koskenniemi. 1990. Finite-state parsing and disambiguation. In *Proc. COLING'90*, volume 2, pages 229–232, Helsinki.

Kimmo Koskenniemi. 1997. Representations and finite-state components in natural language. In *(Roche and Schabes, 1997)*, pages 99–116.

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 Comput. Linguistics*, volume 2099 of *Lecture Notes in Artificial Intelligence*, pages 212–227. Springer-Verlag.

O. Matz, A. Miller, A. Potthoff, W. Thomas, and E. Valkema. 1995. Report on the program AMoRE. Bericht Nr. 9507, Institut für Informatik und Practische Mathematik, Christian-Albrects-Universitt, Kiel.

Robert McNaughton and Seymour Papert. 1968. The syntactic monoid of a regular event. In M.A. Arbib, editor, *Algebraic Theory of Machines, Languages, and Semigroups*, pages 297–312. Academic Press.

Robert McNaughton and Seymour Papert. 1971. *Counter-free Automata*. Research Monograph No. 65. MIT Press.

Yu. T. Medvedev. 1964. On the class of events representable in a finite automaton. In E.F. Moore, editor, *Sequential Machines*, pages 215–227. Addison Wesley.

Mehryar Mohri. 1995. Matching patterns of an automaton. In *Proc. Combinatorial Pattern Matching (CPM'95)*, volume 937 of *LNCS*, pages 286–297, Espoo, Finland. Springer-Verlag.

Jean-Eric Pin. 1986. *Varieties of Formal Languages*. Foundations of Computer Science. North Oxford.

Emmanuel Roche and Yves Schabes, editors. 1997. *Finite-state language processing*. A Bradford Book, MIT Press, Cambridge, MA.

Kai Salomaa and Sheng Yu. 2000. Alternating finite automata and star-free languages. *Theoretical Computer Science*, 234:167–176.

Marcel Paul Schützenberger. 1962. Finite counting automata. *Information and Control*, 5(2):91–107.

Marcel Paul Schützenberger. 1965. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194.

A. Prasad Sistla and Edmund M. Clarke. 1985. The complexity of propositional linear temporal logic. *Journal of ACM*, 32:733–749.

Pasi Tapanainen. 1992. *Äärellisiin automaatteihin perustuva luonnollisen kielen jäsennin*. Licentiate thesis, Department of Computer Science, University of Helsinki, Finland.

Pasi Tapanainen. 1997. Applying a finite-state intersection grammar. In *(Roche and Schabes, 1997)*, pages 311–327.

Nathan Vaillette. 2001. Logical specification of transducers for NLP. In *Finite State Methods in Natural Language Processing 2001 (FSMNLP 2001), ESSLLI Workshop*, pages 20–24, Helsinki.

Moshe Y. Vardi. 1995. Alternating automata and program verification. In *Computer Science Today - Recent Trends and Developments*, volume 1000 of *LNCS*, pages 471–485. Springer-Verlag.

Atro Voutilainen. 1994. *Designing a Parsing Grammar*. Nr. 22 in Publications of the Department of General Linguistics. University of Helsinki.