Compiling Contextual Restrictions on Strings into Finite-State Automata

Anssi Yli-Jyrä and Kimmo Koskenniemi
Department of General Linguistics
P.O. Box 9, FIN-00014 University of Helsinki, Finland
firstname.lastname@helsinki.fi

Abstract

The paper discusses a language operation that we call *context restriction*. This operation is closely associated with *context restriction rules* (Koskenniemi, 1983; Kiraz, 2000), *right-arrow rules* or *implication rules* (Koskenniemi et al., 1992; Voutilainen, 1997) and the *restriction operator* (Beesley and Karttunen, 2003). The operation has been used in finite-state phonology and morphology in certain limited ways. A more general setting involves restricting overlapping occurrences of a center language under context conditions. Recently, star-free regular languages (and all regular languages) have been shown to be closed under context restrictions with such "overlapping centers" (Yli-Jyrä, 2003), but the construction involved is overly complex and becomes impractical when the number of operands grows.

In this paper, we improve this recent result by presenting a more practical construction. This construction is not only simpler but it also leads to a generalization where contexts and centers may appear as conditions at both sides of the implication arrow ($\Rightarrow$): licensing conditions on the right-hand side specify the restriction and triggering conditions on the left-hand side regulate activation of the restriction. One application of the generalization is to facilitate splitting certain context restriction rules in grammars into a conjunction of separate rules.

1 Introduction

There are different definitions for context restriction operations and rules, but they share a common idea that substrings that belong to a so-called *center language* $\mathcal{X}$ are either *accepted* or *rejected* according to the *context* where they occur.[1] A set

[1] In context restriction rules that are used in morphology, the alphabet of the strings consists of same-length correspondences. However, we avoid this complication in the current paper.

of *licensing context conditions* ($\mathfrak{C}_1, \mathfrak{C}_2, \cdots, \mathfrak{C}_n$) is specified, and each of the conditions is a pair of a *left-hand context language* and a *right-hand context language*. The context of an occurrence of the center $\mathcal{X}$ *satisfies* a context condition $\mathfrak{C}_i$ if its left-hand and right-hand contexts belong, respectively, to the left-hand and right-hand context languages. An occurrence is accepted if its context satisfies at least one of the context conditions.

The strings where different occurrences of $\mathcal{X}$ overlap each other are problematic. To treat such a string, the occurrences of the center are divided into those that are *focused* [2] and those that are *unfocused*. The string is included to the language described by context restriction operations and rules if and only if it all the focused occurrences are accepted. However, the existing definitions for context restrictions choose the focused occurrences in different ways. Some definitions for context restrictions focus all the occurrences (Yli-Jyrä, 2003). Some other definitions (related to Karttunen, 1997; Kempe and Karttunen, 1996; Karttunen, 1996) focus, non-deterministically or deterministically, a set of non-overlapping occurrences in such a way that the unfocused occurrences that remain in the string would overlap with the focused ones. There are further definitions that are partition-based (Grimley-Evans et al., 1996; Kiraz, 2000) or do not really work for long occurrences (Kaplan and Kay, 1994), which means that the occurrences cannot overlap each other at all.

Context restriction is a widely useful operation, and it is closely connected to several formalisms:

- In the classical rule formalism for the *two-level morphology*, the centers of context restriction rules are restricted to single character correspondences (Koskenniemi, 1983). Two-level context restriction rules can be compiled into finite-state transducers (FST) according to a suggestion by Ron Kaplan who solved the problem of multiple contexts in 1980's by means of context markers (Karttunen et al., 1987; Kaplan and Kay, 1994).

- Alternative two-level and multi-tiered formalisms have also been proposed (Ritchie et al., 1992; Grimley-Evans et al., 1996; Kiraz, 2000). In these formalisms, the occurrences of the center cannot overlap at all.

- In the framework of Finite State Intersection Grammar (FSIG) (a flavor of finite-state syntax) (Koskenniemi et al., 1992; Yli-Jyrä, 2003), overlapping occurrences can be focused simultaneously because the centers are not necessarily sets of unary strings as it is the case in the classical two-level morphology. In the literature, there are also cases where context restrictions

[2] A focused occurrence corresponds – as a notion – to the occurrence of $\mathcal{X}$ on a particular *application* of the context restriction rule (Karttunen et al., 1987).

could have been used as a shorthand notation for combinations of other operations (Wrathall, 1977; Grimley-Evans, 1997) and to cover **If-Then** functions of (Kaplan and Kay, 1994, p.370), if the operation only had been available as a pre-defined primitive. In these cases, context restriction can be viewed as a general purpose language operation whose arguments can be e.g. context-free languages (Wrathall, 1977; Yli-Jyrä, 2004 (in print)).

- The *replace(ment) operators* (e.g. (Karttunen, 1997; Kempe and Karttunen, 1996; Beesley and Karttunen, 2003) and the *context-dependent rewrite rules* (Kaplan and Kay, 1994; Mohri and Sproat, 1996) are also related to context restrictions, but multiple applications of the replacement / rewriting rules motivate defining restrictions in such a way that simultaneous foci do not overlap each other.

Various flavors of context restrictions differ from each other mainly due to different conceptions on possible foci in the accepted strings. We will now restrict ourselves to the definition where each string is associated with only one set of focused occurrences of the center substrings. In this set, all occurrences of the center language are focused simultaneously and each occurrence must, thus, be accepted. According to this definition, each string of length n has in the worst case $O(n^2)$ focused occurrences, and it is, therefore, not immediately obvious that regular languages are closed under the operation that has this property.

We will now give an exact definition for the flavor of context restriction (context restriction with "overlapping centers") we are concerned with. Let Σ to be the alphabet for building strings. A *context restriction of a center $\mathcal{X}$ in contexts* $\mathfrak{C}_1, \mathfrak{C}_2, \cdots, \mathfrak{C}_n$ is a operation where $\mathcal{X}$ is a subset of Σ^* and each context $\mathfrak{C}_i$, $1 \leq i \leq n$, is of the form $\mathcal{V}_i \_\_ \mathcal{Y}_i$, where $\mathcal{V}_i, \mathcal{Y}_i \subseteq \Sigma^*$. The operation is expressed using a notation

$$\mathcal{X} \Rightarrow \mathcal{V}_1 \_\_ \mathcal{Y}_1, \mathcal{V}_2 \_\_ \mathcal{Y}_2, \ldots, \mathcal{V}_n \_\_ \mathcal{Y}_n \tag{1}$$

and it defines the set of all strings $w \in \Sigma^*$ such that, for every possible $v, y \in \Sigma^*$ and $x \in \mathcal{X}$ for which $w = vxy$, there exists some context $\mathcal{V}_i \_\_ \mathcal{Y}_i$, $1 \leq i \leq n$, where both $v \in \Sigma^*\mathcal{V}_i$ and $y \in \mathcal{Y}_i\Sigma^*$. If all these sets $\mathcal{X}$, $\mathcal{V}_i$ and $\mathcal{Y}_i$ are regular (or star-free regular) then the result will also be regular (resp. star-free regular) (Yli-Jyrä, 2003).

(typo)
$v \in \mathcal{V}_i$
$y \in \mathcal{Y}_i$

The reader should note that, in the current paper, we define context languages $\mathcal{V}_i$ and $\mathcal{Y}_i$, $1 \leq i \leq n$, directly as *total contexts*. [3]

[3] In the literature (e.g. in Beesley and Karttunen, 2003), it is most often assumed that left-hand context languages $\mathcal{V}_i$ of the form Σ^*L and right-hand context languages $\mathcal{Y}_i$ of the form $L\Sigma^*$ can be

In this paper, we present a previously unpublished construction for the language denoted by context restrictions with "overlapping centers". The construction is based on a combination of usual regular operations that are easy to implement in practice. In contrast to various previous compilation methods, our new construction restricts all overlapping occurrences (vs. Kaplan and Kay, 1994; Grimley-Evans et al., 1996), and avoids exponential growth in the size of the expanded formula (vs. Yli-Jyrä, 2003). Our construction resembles the compilation method by Grimley-Evans et al. (1996) and Kiraz (2000). However, our method deals with overlapping occurrences, while theirs assumes a partitioning where the occurrences corresponds to disjoint blocks in the strings. The new method has been communicated to the Xerox group and it has already been adopted – due to its generality and speed – in XFST (Beesley and Karttunen, 2003)[4], a proprietary finite-state compiler, since XFST version 8.3.3. The new construction also generalizes so that it involves *triggering conditions* and *licensing conditions*. The *generalized restriction* has a lot of applications. A part of this paper is devoted to illustration of a possible application that allows for *decomposed context restrictions*.

The paper is structured as follows. The new construction is presented in Section 2 and generalized in Section 3. In Section 4, we describe a special problematic setting where the context restriction has bad state complexity, and then show that the generalized context restriction can be used to split context restrictions into subconstraints that are recognized with smaller automata. We list some areas of further work in Section 5, and conclude in Section 6. The appendix presents a summary of the previous solutions, being of interests only to a portion of the intended audience.

2 New Construction

2.1 Notation

We use the following usual language operations: concatenation (L_1L_2), exponentiation (L^n), concatenation closure (L^*), union ($L_1 \cup L_2$), intersection ($L_1 \cap L_2$) and asymmetric difference ($L_1 - L_2$). We use parenthesis $(,)$ for grouping. When L is a regular language, we denote by $|L|$ the size of a minimal deterministic automaton that recognizes L – this is what we mean by the *state complexity* of L.

The default alphabet for building strings is Σ. Let $\diamond \notin \Sigma$. We use $\diamond$ as a special marker symbol, called a *diamond*, that is not present in the final result. For

simplified by replacing them with L in the notation. This would require prepending and appending Σ^* respectively to $\mathcal{V}_i$ and $\mathcal{Y}_i$ when the meaning of the operation is concerned. We avoid such expansions for the sake of clarity. By doing so, we do not sacrifice generality.

[4]The original XFST version attached to the book by Beesley and Karttunen (2003) would interprete simple and multi-context restrictions under different, mutually inconsistent definitions.

all alphabets M such that $M \cap \Sigma = \emptyset$ we denote the union $\Sigma \cup M$ by Σ_M. By $h_M : \Sigma_M^* \to \Sigma^*$ we denote a homomorphism w.r.t. string concatenation such that it just deletes marker symbols M from the strings. The inverse homomorphism h_M^{-1} obviously inserts symbols M freely in any string position. By $s_{\alpha/L} : \Sigma_{\{\alpha\}}^* \to \Sigma^*$ we denote the substitution that replaces in strings of $\Sigma_{\{\alpha\}}^*$ the symbol $\alpha \notin \Sigma$ with the language $L \subseteq \Sigma^*$.

2.2 Compiling Basic Restrictions

The semantics of a context restriction is formulated in four stages:

(I) We take all possible strings $w \in \Sigma^*$ where there is some focused occurrence of any substring $x \subseteq \mathcal{X}$ and insert a pair of diamonds $\diamond$ into these strings in such a way that the occurrence of x is marked by a preceding diamond $\diamond$ and a following diamond $\diamond$. The obtained set is obviously

$$\Sigma^* \diamond \mathcal{X} \diamond \Sigma^*. \tag{2}$$

Now $h_{\{\diamond\}}(\Sigma^* \diamond \mathcal{X} \diamond \Sigma^*)$ can be interpreted as the set

$$\{w \in \Sigma^* \mid \exists vxy : w = vxy \wedge x \in \mathcal{X}\}.$$

(II) We describe all strings $w = vxy$ where the string pair $v \_\_\_ y$ satisfies some licensing context $\mathcal{V}_i \_\_\_ \mathcal{Y}_i$, and we insert a pair of diamonds $\diamond$ into these strings in such a way that x is marked by a preceding and a following diamond. The obtained set is obviously

$$\cup_{i=1}^n \mathcal{V}_i \diamond \Sigma^* \diamond \mathcal{Y}_i \tag{3}$$

Now $h_{\{\diamond\}}(\cup_{i=1}^n \mathcal{V}_i \diamond \Sigma^* \diamond \mathcal{Y}_i)$ can be interpreted as the set

$$\{w \in \Sigma^* \mid \exists vxy : w = vxy \wedge \exists\, i : 1 \leq i \leq n \wedge v \in \mathcal{V}_i \wedge y \in \mathcal{Y}_i\}.$$

(III) A string $w \in \Sigma^*$ is obviously rejected by the context restriction if and only if it contains some focused occurrence of any $x \in \mathcal{X}$ such that the occurrence does not have a licensing context. If we take all rejected string and add diamonds around an arbitrary x that fails to have a licensing context, we obtain the set:

$$\Sigma^* \diamond \mathcal{X} \diamond \Sigma^* - \cup_{i=1}^n \mathcal{V}_i \diamond \Sigma^* \diamond \mathcal{Y}_i. \tag{4}$$

Now $h_{\{\diamond\}}(\Sigma^* \diamond \mathcal{X} \diamond \Sigma^* - \cup_{i=1}^n \mathcal{V}_i \diamond \Sigma^* \diamond \mathcal{Y}_i)$ can be interpreted as the set

$$\{w \in \Sigma^* \mid \exists vxy : w = vxy \wedge x \in \mathcal{X} \wedge \neg \exists\, i : 1 \leq i \leq n \wedge v \in \mathcal{V}_i \wedge y \in \mathcal{Y}_i\}.$$

(IV) The set (4) consists of all possible rejected strings, with the extra diamonds included. The desired interpretation of the restriction is therefore achieved by deleting the diamonds and taking the complement:

$$\Sigma^* - h_{\{\diamond\}}(\Sigma^* \diamond \mathcal{X} \diamond \Sigma^* - \cup_{i=1}^{n} \mathcal{V}_i \diamond \Sigma^* \diamond \mathcal{Y}_i). \tag{5}$$

This can be interpreted as the set:

$$\{w \in \Sigma^* \mid \neg(\exists vxy : w = vxy \wedge x \in \mathcal{X} \wedge \neg \exists\, i : 1 \leq i \leq n \wedge v \in \mathcal{V}_i \wedge y \in \mathcal{Y}_i)\}$$
$$= \{w \in \Sigma^* \mid \forall vxy : w = vxy \wedge x \in \mathcal{X} \rightarrow \exists\, i : 1 \leq i \leq n \wedge v \in \mathcal{V}_i \wedge y \in \mathcal{Y}_i)\}.$$

The language defined by expression 5 is the language denoted by expression 1. Given this equivalence, we are now able to construct a finite automaton corresponding to expression 1 via usual algorithms on automata, if the argument languages ($\mathcal{X}$, $\mathcal{V}_1$, $\mathcal{Y}_1$, $\mathcal{V}_2$, $\mathcal{Y}_2$, etc.) are regular and given as finite automata.

Those readers who are interested to relate our new construction with previous solutions are directed to the appendix where some earlier solutions are discussed.

3 Generalized Restriction

3.1 Definition

Now we define a new operator $\overset{g\diamond}{\Rightarrow}$, called the *generalized restriction operator*, as follows. Recall that $\mathfrak{W}$ is a Fraktur capital for 'w' and that we used variable 'w' for complete strings. Let $\mathfrak{W}_1, \ldots, \mathfrak{W}_m, \mathfrak{W}'_1, \ldots, \mathfrak{W}'_n$ be subsets of $\Sigma^*(\diamond\Sigma^*)^g$, where $g \in \mathbb{N}$. The expression

$$\mathfrak{W}_1, \mathfrak{W}_2, \ldots, \mathfrak{W}_m \overset{g\diamond}{\Rightarrow} \mathfrak{W}'_1, \mathfrak{W}'_2, \ldots, \mathfrak{W}'_n,$$

denotes the language

$$\Sigma^* - h_{\{\diamond\}}(\cup_{i=1}^{m} \mathfrak{W}_i - \cup_{i=1}^{n} \mathfrak{W}'_i). \tag{6}$$

3.2 Basic Applications

The generalized restriction operation has a potential to express many different kinds of restrictions in ways that are very similar to each other. This flexibility is illustrated by the following examples.

Context restriction Context restriction defined in (1) can be expressed as a generalized restriction as follows:

$$\Sigma^* \diamond \mathcal{X} \diamond \Sigma^* \quad \overset{2\diamond}{\Rightarrow} \quad \mathcal{V}_1 \diamond \Sigma^* \diamond \mathcal{Y}_1,\ \mathcal{V}_2 \diamond \Sigma^* \diamond \mathcal{Y}_2,\ \ldots,\ \mathcal{V}_n \diamond \Sigma^* \diamond \mathcal{Y}_n. \tag{7}$$

Coercion In analogy to the surface coercion rule in the two-level morphology (Koskenniemi, 1983; Kaplan and Kay, 1994; Grimley-Evans et al., 1996), we can define a coercion operation where satisfaction of any *triggering context condition* implies that the focused substrings are drawn from the *licensing center language*. This operation can be defined easily as follows. The expression

$$\mathcal{X}' \Leftarrow \mathcal{V}'_1 \_\_ \mathcal{Y}'_1, \mathcal{V}'_2 \_\_ \mathcal{Y}'_2, \ \ldots, \ \mathcal{V}'_m \_\_ \mathcal{Y}'_m,$$

where the backward arrow indicates that the roles of the sides are exchanged, denotes the language

$$\mathcal{V}'_1 \diamond \Sigma^* \diamond \mathcal{Y}'_1, \ \mathcal{V}'_2 \diamond \Sigma^* \diamond \mathcal{Y}'_2, \ \cdots, \ \mathcal{V}'_m \diamond \Sigma^* \diamond \mathcal{Y}'_m \quad \overset{2\diamond}{\Rightarrow} \quad \Sigma^* \diamond \mathcal{X}' \diamond \Sigma^*.$$

If-Then Kaplan and Kay (1994) defined the following functions:

$$\textit{If-P-then-S}(L_1, L_2) \overset{def}{=} \Sigma^* - L_1(\Sigma^* - L_2)$$
$$\textit{If-S-then-P}(L_1, L_2) \overset{def}{=} \Sigma^* - (\Sigma^* - L_1)L_2.$$

These functions can also be defined very intuitively using generalized restrictions with one diamond:

$$\textit{If-P-then-S}(L_1, L_2) \overset{def}{=} L_1 \diamond \Sigma^* \overset{1\diamond}{\Rightarrow} \Sigma^* \diamond L_2$$
$$\textit{If-S-then-P}(L_1, L_2) \overset{def}{=} \Sigma^* \diamond L_2 \overset{1\diamond}{\Rightarrow} L_1 \diamond \Sigma^*.$$

Nowhere Often we want to say that strings belonging to $\mathcal{X}$ do not occur anywhere in the accepted strings as substrings. This can be expressed as a context restriction $\mathcal{X} \Rightarrow \emptyset \_\_ \emptyset$ or as a generalized restriction $\Sigma^* \mathcal{X} \Sigma^* \overset{0\diamond}{\Rightarrow} \emptyset$.

More than Two Diamonds The number g of diamonds involved in the generalized restriction operation $\overset{g\diamond}{\Rightarrow}$ can also be greater than two. Such extensions can be used to express restrictions on discontinuous parts of the strings, but these possibilities are not discussed in this paper.

3.3 Adding Preconditions

The most appealing property of generalized restrictions is that they contain simultaneously centers and contexts of two different kinds: (i) licensing and (ii) triggering. This allows expressing more complicated rules in a simple and elegant manner:

Context Restriction with Preconditions When we reduce context restrictions into generalized restrictions, the left hand-hand side of the generalized restriction is of the form

$$\mathfrak{W} = \mathcal{V}' \diamond \mathcal{X} \diamond \mathcal{Y}' \text{ where } \mathcal{V}', \mathcal{Y}' = \Sigma^*.$$

The languages $\mathcal{V}'$ and $\mathcal{Y}'$ form a triggering context condition. If we make $\mathcal{V}'$ or $\mathcal{Y}'$ more restrictive, the context restriction focuses only those occurrences whose contexts satisfy the triggering context condition. When the triggering context condition $\mathcal{V}' \_\_\_ \mathcal{Y}'$ is not satisfied by the context of an occurrence, the occurrence is not focused at all and the acceptance of the whole string does not depend on it.

Coercion with Preconditions When we reduce coercions to generalized restrictions, the left hand-hand side of the generalized restriction is of the form

$$\mathcal{V}'_1 \diamond \mathcal{X}_1 \diamond \mathcal{Y}'_1,\ \mathcal{V}'_2 \diamond \mathcal{X}_2 \diamond \mathcal{Y}'_2,\ \cdots,\ \mathcal{V}'_m \diamond \mathcal{X}_m \diamond \mathcal{Y}'_m, \text{where } \mathcal{X}_i = \Sigma^* \text{ for all } 1 \leq i \leq m.$$

Now the sets $\mathcal{X}_i$, where $1 \leq i \leq m$, could also differ from Σ^*. If we make a $\mathcal{X}_i$ more restrictive, an actual context $v \_\_\_ y$, where $v \in \mathcal{V}'_i$ and $y \in \mathcal{Y}'_i$, triggers the coercion on the substring if only if it the focused substring x in the context $v \_\_\_ y$ belongs to set $\mathcal{X}_i$.

Bracketing Restriction Coercion with preconditions can be used to express the meaning of constraints called *bracketing restrictions* (Yli-Jyrä, 2003 (in print)). A bracketing restriction constraint is expressed through the notation

$$\#\mathcal{V}' \_\_\_ \mathcal{Y}'\# \Rightarrow \mathcal{X}',$$

where $\mathcal{V}', \mathcal{Y}', \mathcal{X}' \subseteq \Sigma_{\{\Delta\}}$ denote regular languages. The constraint denotes the language

$$\{w \in \Sigma^* \mid w \in \Delta' \wedge \forall vxy : w = vxy \wedge x \in \Delta' \ \wedge \\ v \in (s_{\Delta/\Delta'}(\mathcal{V}')) \ \wedge \ y \in (s_{\Delta/\Delta'}(\mathcal{Y}')) \longrightarrow x \in (s_{\Delta/\Delta'}(\mathcal{X}'))\},$$

where $\Delta' \subseteq \Sigma^*$ is a bracketed *language* that is substituted for *symbol* Δ. This language can be expressed using a coercion with preconditions as follows:

$$\Delta' \cap \Big((s_{\Delta/\Delta'}(\mathcal{V}')) \diamond \Delta' \diamond (s_{\Delta/\Delta'}(\mathcal{Y}')) \overset{2\diamond}{\Rightarrow} \Sigma^* \diamond (s_{\Delta/\Delta'}(\mathcal{X}')) \diamond \Sigma^* \Big).$$

When the language D is a regular language, such as Δ_d in Section 4.1, every bracketing restriction denotes a regular language. Yli-Jyrä (2003 (in print)) discusses its state complexity and suggests decomposing each restriction into sub-constraints.

Decomposed Context Restriction Context restrictions with preconditions allow us to decompose a context restriction into a set of generalized restrictions whose intersection represents the accepted language. The context restriction

$$\mathcal{X} \Rightarrow \mathcal{V}_1 \_\_\_ \mathcal{Y}_1, \mathcal{V}_2 \_\_\_ \mathcal{Y}_2, \ldots, \mathcal{V}_n \_\_\_ \mathcal{Y}_n,$$

can now be expressed as an intersection of generalized restrictions

$$\begin{array}{rcl} \mathcal{V}'_1 \diamond \mathcal{X} \diamond \mathcal{Y}'_1 & \overset{2\diamond}{\Rightarrow} & \mathcal{V}_1 \diamond \Sigma^* \diamond \mathcal{Y}_1,\ \ldots,\ \mathcal{V}_n \diamond \Sigma^* \diamond \mathcal{Y}_n \\ & \vdots & \\ \mathcal{V}'_m \diamond \mathcal{X} \diamond \mathcal{Y}'_m & \overset{2\diamond}{\Rightarrow} & \mathcal{V}_1 \diamond \Sigma^* \diamond \mathcal{Y}_1,\ \ldots,\ \mathcal{V}_n \diamond \Sigma^* \diamond \mathcal{Y}_n \end{array} \tag{8}$$

such that $\cup_{i=1}^{m} \mathcal{V}'_i \diamond \mathcal{X} \diamond \mathcal{Y}'_i = \Sigma^* \diamond \mathcal{X} \diamond \Sigma^*$.

4 A Blow-Up Problem with Context Restrictions

In some practical applications, the languages described by context restrictions have bad state complexity. In the sequel, we will present the background of this problem and then show how it could be solved using decomposed context restrictions. The solution represents each context restriction by means of an intersection of simpler languages. Such an intersection corresponds to a direct product of small automata, where the direct product can be computed lazily, on demand. The improved representation is more compact and better organized, which facilitates efficient application of context restrictions.

4.1 The Background

Bracketed Finite-State Intersection Grammars An approach for natural language parsing and disambiguation based on regular languages as constraints was proposed in (Koskenniemi et al., 1992). It formed the theoretical basis for a Finite-State Intersection Grammar (FSIG) for English (Voutilainen, 1997). This particular FSIG has, however, suffered from parsing difficulties (Tapanainen, 1997). Recently, the parsing approach has been developed into new kinds of FSIG grammars that could be called Bracketed FSIGs (Yli-Jyrä, 2003; 2003 (in print); 2004 (in print); 2004a). In these FSIGs, a great deal of the state complexity of the grammar derives from balanced bracketing that is to be validated by means of finite-state constraints.

Marking the Sentence Structure In the FSIG approach, the parses for natural language sentences are represented as annotated sentences. An annotated sentence might be a sequence of multi-character symbols, including word tokens and correctly inserted part-of-speech tags and brackets, as in the following:

```
the DET man N [ who PRON walked V PAST on PREP the DET
street N ] was V happy A
```

or

```
this PRON is V the DET dog N [ that PRON chased V the
DET cat N ] [ that PRON killed V the DET mouse ].
```

In these examples, the brackets are used to mark a part of the clause structure. In some Bracketed FSIGs, the brackets mark very detailed constituency (Yli-Jyrä, 2003 (in print)) or dependency structures (Yli-Jyrä, 2004 (in print)). Nevertheless, brackets can be used economically (cf. Koskenniemi et al., 1992; Yli-Jyrä, 2003 (in print), 2004a) so that the depth of nested brackets remains small in practice.

The Language with Balanced Brackets In all the annotated sentences of FSIG, the brackets belonging to a class of left "super" brackets is balanced with the brackets belonging to the corresponding class of right "super" brackets. This property can be expressed as follows: Let B_L and B_R be respectively a class of left and right "super" brackets in the grammar. There may be other, non-"super" brackets that are not balanced, but are closed (or opened) with a "super" bracket. The set $\Delta_d \subseteq \Sigma^*$ contains all bracketed strings whose "super" brackets are balanced w.r.t. these bracket classes, and where the "super" brackets have at most d nested levels (level 0 = no brackets). This language is derived inductively as follows:

$$\Delta_d = \begin{cases} (\Sigma - B_L - B_R)^* & \text{if } d = 0 \\ (\Delta_{d-1} \cup (\ B_L\ \Delta_{d-1}\ B_R\ ))^* & \text{if } d > 0 \end{cases}$$

Voutilainen (1997) uses in his FSIG for English a special pre-defined language *dots* '**...**' (equivalent to $\boxed{\cdots}^d$ in Yli-Jyrä, 2003) to refer to arbitrary strings with balanced bracketing, where the bracketing marks boundaries of center embedded clauses. It is obtained as the language $\Delta_1 - \Sigma^*$ @@ Σ^* if we let $B_L = \{$@<$\}$ and $B_R = \{$@>$\}$.

Typical Context Restriction Rules Typical FSIG rules express contextual restrictions on features within clauses by requiring the presence of other syntactic functions and structures to the left and/or to the right. Normally only features within the same clause will be used as a context, but in case of relative pronouns

and conjunctions the required features might be after or before the clause boundary (cf. e.g. Voutilainen, 1997).

Most FSIG rules are written in such a way that they apply to deeper clauses as well as to top-level clauses. This is made possible by means of a language of balanced bracketings – denoted by '**...**', $\boxed{\cdots}^d$, or Δ_d – that is pre-defined in all FSIG frameworks. For example, the rule

$$\texttt{@SUBJ} \;\Rightarrow\; \Sigma^* \;\texttt{VFIN}\; \boxed{\cdot\cdot}^d \_\_\_ \Sigma^*, \quad \Sigma^* \_\_\_ \boxed{\cdot\cdot}^d \;\texttt{VFIN}\; \Sigma^*$$

requires that a subject feature (`@SUBJ`) is allowed only in clauses where one can also find a feature indicating a finite verb (`VFIN`). The *dotdot* $\boxed{\cdot\cdot}^d$ denotes the set of strings that are in the same clause. It is defined by the formula $\boxed{\cdot\cdot}^d = \boxed{\cdots}^d - \boxed{\cdot}^d \;\texttt{@/}\; \boxed{\cdot}^d$, where `@/` is a multi-character symbol that is used in bracketed strings to separate adjacent clauses at the same bracketing level.

4.2 Large Compilation Results

The size of the automata obtained from context restriction rules has turned out to be an obstacle for large-scale grammar development (Voutilainen, 1997). Usually we are not able to combine many context restriction rules into a single automaton. Therefore, the compiled grammar must be represented as a lazily evaluated combination (intersection) of individual context restriction rules.

Unfortunately, even separate context restriction rules can be too complex to be compiled into automata. When the center $\mathcal{X}$ or a context $\mathcal{C}_i$ is defined using the $\boxed{\cdots}^d$ or Δ_d language, the automaton obtained from a context restriction seems to grow in the worst case exponentially according to the parameter d. This effect can be perhaps most easily understood as follows: If the automaton that enforces a context restriction on unbracketed strings has k states, the automaton that enforces the restriction on one-level bracketings needs, on one hand, $O(k)$ states for keeping track of the restriction on the top level and, on the other hand, $O(k)$ states for keeping track of the restriction inside bracketed embeddings at each $O(k)$ possible states of the the top level automaton. In practice, this means that the worst-case state complexity of this automaton is in $O(k^2)$. Furthermore, it seems that the state complexity will grow exponentially according to d.

Due to these observations on the state complexity of context restriction rules, the second author of this paper proposed that we should try to split the rules into sub-rules each of which takes care of a different bracketing level[5].

[5]The idea of splitting FSIG rules into separate levels is due to the second author (Koskenniemi) who communicated it privately to the first author several years before we learned to do it constructively.

4.3 Decomposition w.r.t. Bracketing Level

We will now present a new compilation method that decomposes an arbitrary context restriction involving Δ_d into a set of generalized restrictions. Each of these generalized restrictions can then be compiled separately into finite automata.

One of the underlying assumptions in FSIGs is that the accepted strings belong to the set Δ_d (or $\boxed{\cdots}^d$). For this reason, we can replace each context restriction

$$\mathcal{X} \Rightarrow \mathcal{V}_1 \_\_ \mathcal{Y}_1, \mathcal{V}_2 \_\_ \mathcal{Y}_2, \ldots, \mathcal{V}_n \_\_ \mathcal{Y}_n$$

with

$$\Delta_d \ \cap \ \ (\mathcal{X} \Rightarrow \mathcal{V}_1 \_\_ \mathcal{Y}_1, \mathcal{V}_2 \_\_ \mathcal{Y}_2, \ldots, \mathcal{V}_n \_\_ \mathcal{Y}_n). \tag{9}$$

Observe that the prefixes of balanced bracketings Δ_d belong to the language $P = \cup_{i=0}^{d} \Delta_d (B_L \Delta_d)^i$. All the other prefixes in Σ^* are in the set $\Sigma^* - P$, but they are not possible prefixes for the strings in Δ_d. Accordingly, we can replace Formula (9) with an intersection of generalized restrictions as follows:

$$\cap_{i=0}^{d} L_i, \tag{10}$$

where the languages L_i are defined by the formula

$$L_i = \Delta_d \cap ((\Delta_d (B_L \Delta_d)^i) \diamond \mathcal{X} \diamond \Sigma^* \overset{2\diamond}{\Rightarrow} \mathfrak{W}'), \text{ where } \mathfrak{W}' = \cup_j^n (\mathcal{V}_j \diamond \Sigma^* \diamond \mathcal{Y}_j).$$

The language L_i, $0 \leq i \leq d$, restricts occurrences of $\mathcal{X}$ that start at the bracketing level i. Each L_i can be compiled into a separate constraint automaton and the intersection of the languages of these automata will be computed lazily during FSIG parsing.

Our hypothesis is that the obtained new lazy representation (sometimes such representations are called *virtual networks*) for the decomposed context restriction (10) will be substantially smaller than the single automaton that results from Formula (9). This hypothesis motivates the following experiment.

4.4 An Experiment

We carried out a small experiment with different representations of context restriction rules in FSIG. In the experiment we investigated the following possibilities:

- each rule corresponds to a separate constraint language $R, S, \ldots$,
- the rules are combined into a single, big constraint language $R \cap S \cap \cdots$,

- each rule is decomposed into $d+1$ separate languages $R_0, R_1, \ldots, R_d$ as proposed in Section 4.3,
- languages R_i, $S_i, \ldots$ for each bracketing level i, $0 \leq i \leq d$, are combined into a big constraint language $R_i \cap S_i \cap \cdots$.

The Set of Rules Interesting rules (Voutilainen, 1997) taken from a full-scale grammar would have been too complicated to be investigated here. The following context restriction rules, expressing simple generalizations about the presence of syntactic categories, were used in the experiment:

$$\Delta_d \cap (\ \texttt{IOBJ} \ \Rightarrow\ \Sigma^*\ \texttt{OBJ}\ \Delta_d \_\_\_ \Sigma^*,\ \Sigma^* \_\_\_ \Delta_d\ \texttt{OBJ}\ \Sigma^*) \qquad (R)$$

$$\Delta_d \cap (\ \texttt{OBJ} \ \Rightarrow\ \Sigma^*\ \texttt{SUBJ}\ \Delta_d \_\_\_ \Sigma^*,\ \Sigma^* \_\_\_ \Delta_d\ \texttt{SUBJ}\ \Sigma^*) \qquad (S)$$

$$\Delta_d \cap (\ \texttt{SUBJ} \ \Rightarrow\ \Sigma^*\ \texttt{VFIN}\ \Delta_d \_\_\_ \Sigma^*,\ \Sigma^* \_\_\_ \Delta_d\ \texttt{VFIN}\ \Sigma^*) \qquad (T)$$

These rules correspond to automata that are similar to each other up to relabeling of certain transitions. One should note, however, that rules R and S (and rules S and T) have more features in common than rules R and T.

The Size of Automata The state complexity of the rules R, S, and T grows exponentially according to d, the bound for nested brackets. This is shown in Table 1. For comparison, the language R' is defined as $(\texttt{IOBJ} \Rightarrow \Sigma^*\ \texttt{OBJ}\ \Delta_d \_\_\_ \Sigma^*,\ \Sigma^* \_\_\_ \Delta_d\ \texttt{OBJ}\ \Sigma^*)$.

d	$\lvert\Delta_d\rvert$	$\lvert R'\rvert$	$\lvert R\rvert$	$3\lvert R\rvert + 3$	3^d
0	1	3	3	12	9
1	2	9	12	39	36
2	3	27	39	120	108
3	4	81	120	363	324
4	5	243	363	1092	972

The last column should read: $\underline{\underline{3^d 12}}$, 12, 36, 108, 324, 972

Table 1: The state complexity of language R grows exponentially according to d.

Assuming that $d = 2$, we constructed automata for each of the languages R, S and T and splitted them for each bracketing level in order to obtain languages R_i, S_i and T_i for all i, $0 \leq i \leq d$. These automata were then combined in different ways in order to see how the sizes of automata differ in the four theoretical possibilities. The results are shown in Table 2. It shows that combinations of full rules grow substantially faster than combinations of rules decomposed with respect to the bracketing level. When we decompose the language R w.r.t. the bracketing

L	$\|L\|$	$\|L_0\|$	$\|L_1\|$	$\|L_2\|$	sum
R	39	9	7	5	21
S	39	9	7	5	21
T	39	9	7	5	21
$R \cap S$	258	18	13	8	39
$R \cap T$	819	27	19	11	57
$R \cap S \cap T$	1884	36	25	14	75

Table 2: State complexity: the rules without decomposition compared to rules that have been decomposed w.r.t. bracketing levels.

depth using the formula (10), we see in Table 3 that the state complexity of the top-most component (R_0) grows linearly to d and $|\Delta_d|$.

	$d=0$	$d=1$	$d=2$	$d=3$	d
$\|\Delta_d\|$	1	2	3	4	$(d+1)$
$\|R_0\|$	3	6	9	12	$3(d+1)$

Table 3: The state complexity of R_0 grows linearly to the parameter d.

Application Relevance For purposes of FSIG parsing, it would be nice if we could compute a minimal deterministic automaton that recognizes the intersection of all rules in the grammar. Because this cannot be done in practice, a parsing algorithm based on backtracking search has been used earlier (Tapanainen, 1997) in addition to automata construction algorithms. The search algorithm operated mostly in a left-to-right fashion.

We observe that $R_i \cap S_i \cap T_i$ have substantially smaller state complexity than $R \cap S \cap T$. This is an important finding, having applications in FSIG parsing. So far, it has not been feasible to apply all the rule automata one by one with a kind of word lattice called a *sentence automaton* (Tapanainen, 1997) by computing successive direct products of automata and minimizing the results. This might become feasible when rules have been decomposed with the current method. Decomposition of constraint restrictions w.r.t. bracketing levels apparently facilitates new techniques (Yli-Jyrä, 2004b) where the parsing proceeds partly in a bottom-up or top-down fashion rather than only in a left-right fashion.

5 Further Work

The definition of context restriction used in this paper is not the only possibility. Further research in this area is still needed due to the following reasons:

- It is not obvious whether the current flavor for the context restriction is practical if we extend context restrictions by attaching weights to contexts and centers.
- It is our experience that, for real context restrictions rules, different definitions for the operation may result in equivalent languages. Understanding when the results coincide might have practical relevance.
- It is an open question whether the proposed or the other definitions for context restrictions have a more natural interpretation when they are not interchangeable but yield different results.

While the current results on decomposed context restrictions significantly improve our possibilities to combine large portions of the original FSIG grammars into singe automata, there is still place for further research that is related to methods that organize the computation of the lazy intersection in an efficient manner.

Generalized restriction with multiple diamonds has many useful applications that remain to be studied later.

6 Conclusion

This paper discussed the definition and the representation of the context restriction operation. In particular, an alternative compilation method for generalized context restriction operation was given, with immediate applications to arrangement of constraint automata in finite-state parsers that apply multiple automata to the input.

The main result of this paper is that context restrictions can itself be restricted to be applied only in certain contexts, which means that there are two kinds of contexts that can occur in one rule: those that trigger the restriction and those that license the restricted occurrences. This observation can be used, on one hand, in defining different kinds of operations and, on the other hand, to split large context restrictions into components that can be compiled separately.

The applicability of these results are not restricted to the formalisms where so-called context restriction rules are used. The general context restriction is a useful operation that is derived from the relative complement of languages, and it allows expressing complex situations in an intuitive manner.

Acknowledgments

We are grateful to L. Karttunen, L. Carlson, A. Kempe, S. Wintner and Y. Cohen-Sygal for useful discussions, and anonymous ACL and COLING referees for critical comments. The new constructions, the experiments and the comparisons are due to the first author, whose work was funded by NorFA under his personal Ph.D. scholarship (ref.nr. 010529).

Appendix: Some Previous Solutions

An Approach that Does not Use Transducers

A Well Known Special Case There is a well known formula (Koskenniemi, 1983, p.106; Kaplan and Kay, 1994, p.371)[6] that compiles simple context restrictions with "overlapping centers":

$$\Sigma^* - ((\Sigma^* - \mathcal{V}_1)\mathcal{X}\Sigma^* \cup \Sigma^*\mathcal{X}(\Sigma^* - \mathcal{Y}_1))\,. \tag{11}$$

Compound Restriction Operations with Two Contexts The following compilation method[7] covers all 2-context cases:

$$\begin{array}{lll} \Sigma^*-(\ \ \Sigma^*\mathcal{X}((\Sigma^* - \mathcal{Y}_1) \cap (\Sigma^* - \mathcal{Y}_2)) & & \text{“at least } \mathcal{Y}_1 \text{ and } \mathcal{Y}_2 \text{ fail”} \\ \quad \cup (\Sigma^* - \mathcal{V}_1)\mathcal{X}(\mathcal{Y}_1 - \mathcal{Y}_2) & & \text{“at least } \mathcal{V}_1 \text{ and } \mathcal{Y}_2 \text{ fail”} \\ \quad \cup (\Sigma^* - \mathcal{V}_2)\mathcal{X}(\mathcal{Y}_2 - \mathcal{Y}_1) & & \text{“at least } \mathcal{V}_2 \text{ and } \mathcal{Y}_1 \text{ fail”} \\ \quad \cup ((\Sigma^* - \mathcal{V}_1) \cap (\Sigma^* - \mathcal{V}_2))\mathcal{X}(\mathcal{Y}_1 \cap \mathcal{Y}_2)\ \ ) & & \text{“only } \mathcal{V}_1 \text{ and } \mathcal{V}_2 \text{ fail”} \end{array} \tag{12}$$

It is possible to show that the last line of (12) can be simplified without changing the meaning of the whole formula:

$$\Sigma^*-(\ldots \cup ((\Sigma^* - \mathcal{V}_1) \cap (\Sigma^* - \mathcal{V}_2))\mathcal{X}\Sigma^*\ \ ) \quad \text{“at least } \mathcal{V}_1 \text{ and } \mathcal{V}_2 \text{ fail”} \tag{13}$$

This modified formula is a special case of (14):

[6] Grimley-Evans et al., 1996, p.458, quote Kaplan and Kay imprecisely, suggesting that this formula for simple context restrictions does not work when context language portions overlap with portions of the center language.

[7] This formula is given in a slightly different form (with **If-Then** functions) on the web page "Operators in XFST, FSA Utilities and FSM – A synopsis. Explanation of operators for FSA Utilities.". This page has been available since year 2003 at the location `http://cs.haifa.ac.il/~shuly/teaching/03/lab/fst.html`. In addition to Jason Eisner who made the first version, many anonymous scholars have worked on this page and it is not obvious who contributed the additional sections. In 2003, Yli-Jyrä discussed with Shuly Wintner about the author of the context restriction section. That section can probably be attributed to Dale Gerdemann in Tübingen.

The General Case Yli-Jyrä (2003) generalized the latter formula (13) independently to arbitrary number of contexts. The language denoted by the context restriction with "overlapping centers" is obtained with the formula[8]

$$\Sigma^* - \bigcup_{t_1} \cdots \bigcup_{t_n} \left(\bigcap_{i=1}^{n} \phi(t_i, \mathcal{V}_i) \right) \mathcal{X} \left(\bigcap_{i=1}^{n} \phi(\overline{t_i}, \mathcal{Y}_i) \right), \tag{14}$$

where $t_1, t_2, \ldots, t_n$ are Boolean variables and the function $\phi : \{\mathbf{true}, \mathbf{false}\} \times 2^{\Sigma^*} \rightarrow 2^{\Sigma^*}$ is defined in such a way that $\phi(q, Q)$ returns $\Sigma^* - Q$ if q is **true**, and Σ^* otherwise. In practice, when n is small, Formula (14) is very efficient. However, when n grows, the expansion of the formula results in an exponentially growing regular expression.

To derive (14), consider the situation where a context $v \_\_\_\_ y$ does not satisfy any licensing context condition. It equals to $\bigwedge_{i=1}^{n} v \notin \mathcal{V}_i \vee y \notin \mathcal{Y}_i$. The disjunction $v \notin \mathcal{V}_i \vee y \notin \mathcal{Y}_i$ is true if and only if the formula $t_i \rightarrow v \notin \mathcal{V}_i \wedge \overline{t_i} \rightarrow y \notin \mathcal{Y}_i$ is satisfiable (i.e. true for some value of t_i). Thus, the formula $\bigwedge_{i=1}^{n} v \notin \mathcal{V}_i \vee y \notin \mathcal{Y}_i$ can rewritten as the following formula:

$$\exists t_1 \ldots t_n : \left(\bigwedge_{i=1}^{n} t_i \rightarrow v \notin \mathcal{V}_i \wedge \overline{t_i} \rightarrow y \notin \mathcal{Y}_i \right) \Leftrightarrow \bigvee_{t_1} \cdots \bigvee_{t_n} \left(\bigwedge_{i=1}^{n} t_i \rightarrow v \notin \mathcal{V}_i \wedge \overline{t_i} \rightarrow y \notin \mathcal{Y}_i \right)$$
$$\Leftrightarrow \bigvee_{t_1} \cdots \bigvee_{t_n} \left(v \in \bigcap_{i=1}^{n} \phi(t_i, \mathcal{V}_i) \right) \wedge \left(y \in \bigcap_{i=1}^{n} \phi(\overline{t_i}, \mathcal{Y}_i) \right).$$

When the failure condition is in this form, it is easy to see how it has been used to obtain Formula (14).

Kaplan and Kay's Approach

In the two-level model of morphology (Koskenniemi, 1983, p.106), centers of context restrictions are normally of length of one symbol (of a symbol pair alphabet). Compilation of this special case with arbitrary number of contexts has been solved with aid of marker symbols (Karttunen et al., 1987; Kaplan and Kay, 1994). This solution was originally presented using a pair symbol alphabet and rational relations and it defined such same-length relations that were used in two-level morphology.

In the following, we will reformulate Kaplan and Kay's method in the domain of regular languages (rather than in the domain of same-length relations). The one-context restriction $\mathcal{X}_{\Sigma_M} \Rightarrow_{\Sigma_M} \mathcal{V}_{\Sigma_M} \_\_\_\_ \mathcal{Y}_{\Sigma_M}$ for arguments $\mathcal{X}_{\Sigma_M}, \mathcal{V}_{\Sigma_M}, \mathcal{Y}_{\Sigma_M} \subseteq$

[8] To make the current presentation more coherent, the original formula of Yli-Jyrä (2003) is turned here "up-side down" by an application of DeMorgan's law.

Σ_M^* is used as a primitive operation, which is interpreted as the language $\Sigma_M^* - ((\Sigma_M^* - \mathcal{V}_{\Sigma_M})\mathcal{X}_{\Sigma_M}\Sigma_M^* \cup \Sigma_M^*\mathcal{X}_{\Sigma_M}(\Sigma_M^* - \mathcal{Y}_{\Sigma_M}))$.

The Original Method Kaplan and Kay's method allocates 2 marker symbols for each context C_i. These marker symbols form the set $M = \{\langle_i | 1 \leq i \leq n\} \cup \{\rangle_i | 1 \leq i \leq n\}$. A context restriction with n contexts is compiled as

$$h_M(\ (\cup_{i=1}^{n}\ (\mathcal{X} \Rightarrow_{\Sigma_M} \Sigma_M^* \langle_i \_\_\_ \rangle_i \Sigma_M^*))^* \\ \cap\ \cap_{i=1}^{n}(\langle_i \Rightarrow_{\Sigma_M} h_M^{-1}(\mathcal{V}_i) \_\_\_ \Sigma_M^*) \cap (\rangle_i \Rightarrow_{\Sigma_M} \Sigma_M^* \_\_\_ h_M^{-1}(\mathcal{Y}_i))). \quad (15)$$

This method works, indeed[9], only if $\mathcal{X} \subseteq \Sigma$. For example, if $\mathcal{X} = \{aa\} \subseteq \Sigma^*$, the language described by the sub-formula $(\mathcal{X} \Rightarrow_{\Sigma_M} \Sigma_M^* \langle_i \_\_\_ \rangle_i \Sigma_M^*)$ contains all unary strings Σ, and thus (15) yields the universal language Σ^* regardless of the licensing context conditions.[10]

An Improved Method A slightly more general solution, where $\mathcal{X}$ has the limitation $\mathcal{X} \subseteq \Sigma^*\Sigma$, is

$$h_M((\Sigma^* - \Sigma^*\mathcal{X}\Sigma^*)((\cup_{i=1}^{n} \langle_i \mathcal{X} \rangle_i)\ \ (\Sigma^* - \Sigma^*\mathcal{X}\Sigma^*))^* \\ \cap \cap_{i=1}^{n}(\langle_i \Rightarrow_{\Sigma_M} h_M^{-1}(\mathcal{V}_i) \_\_\_ \Sigma_M^*) \cap (\rangle_i \Rightarrow_{\Sigma_M} \Sigma_M^* \_\_\_ h_M^{-1}(\mathcal{Y}_i))). \quad (16)$$

This improvement is closely related to the replace operator (Karttunen, 1997; Kempe and Karttunen, 1996)[11]. It handles all context restrictions with "non-overlapping centers", but works with "overlapping centers" in a way that differs from our definition (1).

Crucial difference There are examples of context restrictions that can be used to test different definitions and differentiate them from each other. For example,

$$a\Sigma^*b \Rightarrow \Sigma^*c \_\_\_ \Sigma^*, \Sigma^* \_\_\_ d\Sigma^*$$

[9] From Kaplan and Kay, 1994, p.369, one might be able to read that the limitation $\mathcal{X} \subseteq \Sigma$ is inessential.

[10] This resembles an XFST regular expression `[[a => b _ c] | [a => d _ e]]*` by Beesley and Karttunen (2003, p. 65). That cannot be seen as a compilation approach for a context restriction of the form `[a => b _ c, d _ e]`, because it is similarly defective if `a`,`b`,`c`,`d` and `e` are constants denoting arbitrary regular languages.

[11] This method was also related although not precisely identifiable with the obsolete implementation of the restriction operator in some earlier XFST versions (before v.8.3.0). According to Kempe (2004, priv.comm.), e.g. an XFST regular expression `[? - %@]* & [X => L1 _ R2, ..., Ln _ Rn]`, where `%@` is a special symbol not occurring in `X`,`L1`,`R1`,...,`Ln`,`Rn`, would have been compiled in such a way that it corresponds to `[[? - %@]*.o.[X -> %@ || L1 _ R2, ..., Ln _ Rn].o. ~[?* X ?*]].u`. This is *roughly* the method that was used in earlier XFST versions.

demonstrates that the improved method is not equivalent to our definition (1): the result of Formula (16) accepts e.g. strings $caab$, $abdb$, $abbd$, $acab$ while the result of Formula (5) rejects them. Here, the improved method (16) produced a bigger result (7 states) than our method (4 states). In the FSIG framework (Koskenniemi et al., 1992), one can easily find more restrictions rules with "overlapping centers". Some of them would produce different results if compiled using these alternative methods (cf. Yli-Jyrä, 2003).[12]

A Partition-Based Approach

The Original Method Grimley-Evans et al. (1996) implemented a morphological grammar system that involved context restriction rules. This grammar system is partition-based, which means that possible strings (or lexical/surface correspondences) inside the system are sequences of element substrings $E \subseteq \Sigma^*$ (or lexical/surface correspondences) that are separated with a separator $\sigma \notin \Sigma$. The set of all possible sequences that are filtered with context restriction rules is, thus, $\sigma(E\sigma)^*$. The center language $\mathcal{X}$ is a subset of E. Each context restriction focuses only occurrences that are complete elements substrings. When we restrict ourselves to strings instead of correspondences, this compilation method can be formulated as

$$\sigma(E\sigma)^* - s_{\diamond/\sigma\mathcal{X}\sigma}(h^{-1}_{\{\sigma\}}(\Sigma^* \diamond \Sigma^* - \bigcup_{i=1}^{n} \mathcal{V}_i \diamond \mathcal{Y}_i)). \tag{17}$$

A Non-Partition-Based Variant Formula (17) is closely related to our construction (5). However, sequences of element substrings in the partition-based system is an unnecessary complication when the restriction operates with languages rather than regular relations. We can simplify Formula (17) by eliminating this feature. The simplification is formulated as follows:

$$\Sigma^* - s_{\diamond/\mathcal{X}}(\Sigma^* \diamond \Sigma^* - \bigcup_{i=1}^{n} \mathcal{V}_i \diamond \mathcal{Y}_i). \tag{18}$$

Formula (18) captures the definition (1) and it can be seen, therefore, as an optimization for Formula (5).

[12] The rule compiler used by Voutilainen (1997) was implemented by Pasi Tapanainen in Helsinki, and it produced in 1998 results that seem to coincide with our definition (1). The method used in the compiler has not been published, but according to Tapanainen (1992) he has used a transducer-based method for compiling implication rules.

References

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

Grimley-Evans, Edmund. 1997. Approximating context-free grammars with a finite-state calculus. In *Proc. ACL 1997*, pages 452–459. Madrid, Spain.

Grimley-Evans, Edmund, George Anton Kiraz, and Stephen G. Pulman. 1996. Compiling a partition-based two-level formalism. In *Proc. COLING 1996*, vol. 1, pages 454–59.

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

Karttunen, Lauri. 1996. Directed replacement. In *Proc. ACL 1996*, pages 108–115. Santa Cruz, CA, USA.

Karttunen, Lauri. 1997. The replace operator. In E. Roche and Y. Schabes, eds., *Finite-State Language Processing*, chap. 4, pages 117–147. Cambridge, MA, USA: A Bradford Book, the MIT Press.

Karttunen, Lauri, Kimmo Koskenniemi, and Ronald M. Kaplan. 1987. A compiler for two-level phonological rules. Report CSLI-87-108, Center for Study of Language and Information, Stanford University, CA, USA.

Kempe, André and Lauri Karttunen. 1996. Parallel replacement in finite state calculus. In *Proc. COLING 1997*, vol. 2, pages 622–627. Copenhagen Denmark.

Kiraz, George Anton. 2000. Multitiered nonlinear morphology using multitape finite automata: A case study on Syriac and Arabic. *Computational Linguistics* 26(1):77–105.

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

Koskenniemi, Kimmo, Pasi Tapanainen, and Atro Voutilainen. 1992. Compiling and using finite-state syntactic rules. In *Proc. COLING 1992*, vol. 1, pages 156–162. Nantes, France.

Mohri, Mehryar and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. In *Proc. ACL 1996*, pages 231–238. Santa Cruz, CA, USA.

Ritchie, Graeme D., Graham J. Russel, and Alan W. Black. 1992. *Computational Morphology: Practical Mechanisms for the English Lexicon*. Cambridge, MA, USA: A Bradford Book, the MIT Press.

Tapanainen, Pasi. 1992. Äärellisiin automaatteihin perustuva luonnollisen kielen jäsennin. Licentiate thesis C-1993-07, Department of Computer Science, University of Helsinki, Helsinki, Finland.

Tapanainen, Pasi. 1997. Applying a finite-state intersection grammar. In E. Roche and Y. Schabes, eds., *Finite-State Language Processing*, chap. 10, pages 311–327. Cambridge, MA, USA: A Bradford Book, the MIT Press.

Voutilainen, Atro. 1997. Designing a (finite-state) parsing grammar. In E. Roche and Y. Schabes, eds., *Finite-State Language Processing*, chap. 9, pages 283–310. Cambridge, MA, USA: A Bradford Book, the MIT Press.

Wrathall, Celia. 1977. Characterizations of the Dyck sets. *RAIRO – Informatique Théorique* 11(1):53–62.

Yli-Jyrä, Anssi. 2003. Describing syntax with star-free regular expressions. In *Proc. EACL 2003*, pages 379–386. Agro Hotel, Budapest, Hungary.

Yli-Jyrä, Anssi. 2003 (in print). Regular approximations through labeled bracketing (revised version). In G. Jäger, P. Monachesi, G. Penn, and S. Wintner, eds., *Proc. FGVienna, The 8th conference on Formal Grammar, Vienna, Austria 16–17 August, 2003*, CSLI Publications Online Proceedings. Stanford, CA, USA: CSLI Publications.

Yli-Jyrä, Anssi. 2004a. Axiomatization of restricted non-projective dependency trees through finite-state constraints that analyse crossing bracketings. In G.-J. M. Kruijff and D. Duchier, eds., *Proc. Workshop of Recent Advances in Dependency Grammar*, pages 33–40. Geneva, Switzerland.

Yli-Jyrä, Anssi. 2004b. Simplification of intermediate results during intersection of multiple weighted automata. In M. Droste and H. Vogler, eds., *Weighted Automata: Theory and Applications, Dresden, Germany*, no. TUD-FI04-05 — May 2004, ISSN-1430-211X in Technische Berichte der Fakultät Informatik, pages 46–48. D-01062 Dresden, Germany: Techniche Universität Dresden.

Yli-Jyrä, Anssi. 2004 (in print). Approximating dependency grammars through regular string languages. In *Proc. CIAA 2004. Ninth International Conference on Implementation and Application of Automata. Queen's University, Kingston, Canada*, Lecture Notes in Computer Science. Springer-Verlag.

[Corrections added by Anssi Yli-Jyrä in June 2005]