Compiling Simple Context Restrictions with Nondeterministic Automata

Anssi Yli-Jyrä
The Department of Modern Languages, PO Box 3, 00014 University of Helsinki, Finland
anssi.yli-jyra@helsinki.fi

Abstract

This paper describes a non-conventional method for compiling (phonological or morpho-syntactic) context restriction (CR) constraints into non-deterministic automata in finite-state tools and surface parsing systems. The method reduces any CR into a simple one that constraints the occurrences of the empty string and represents right contexts with co-determististic states. In cases where a fully deterministic representation would be exponentially larger, this kind of *inward* determinism in contexts can bring benefits over various De Morgan approaches where full determinization is necessary. In the method, an accepted word gets a unique path that is a projection of a ladder-shaped structure in the context recognizer. This projection is computed in time that is polynomial to the number of context states. However, it may be difficult to take advantage of the method in a finite-state library that coerces intermediate results into canonical automata and whose intersection operation assumes deterministic automata.

1 Introduction

Context restriction (CR) constraints and the related extended regular expression operator (⇒) are included in some widely used finite state compilers (such as XFST, SFST, and FOMA) and is a standard part of Two Level Morphology (Koskenniemi, 1983). In addition, context-sensitive rewriting (e.g. XFST replace rules) have an inherent connection to CR constraints and their implementation can be based on them (Yli-Jyrä, 2008a). Optimized CR compilation methods can thus bring advantage to a wide range of applications.

The current work presents a complement-free method that has some advantages and disadvantages in comparison with the commonly used De Morgan implementations of the inherent universal quantification of CRs. It expresses the universal quantification positively, by recognizing ladder-shaped structures between deterministic left contexts and co-deterministic right contexts and by projecting them to accepted words. The complexity of the method is analyzed here, but a fuller evaluation remains for further work.

1.1 The Use of CR Constraints

Let Σ be the (pair symbol) alphabet over which all the words are defined. A *context restriction (CR)* constraint checks the occurrences of a pattern in the words. For example, a phonological constraint (Koskenniemi, 1983) such as

$$\texttt{p:m} \Rightarrow \Sigma^*\texttt{n:m}\,\_\,\Sigma^*, \qquad (1)$$

specifies a formal language $L \subseteq \Sigma^*$ where the (pair) symbol `p:m` may occur only when immediately preceded by the symbol `n:m`. The left hand side (`p:m`) describes the constrained *pattern* while the right hand side ($\Sigma^*\texttt{n:m}\,\_\,\Sigma^*$) specifies the *contexts* to which the pattern occurrences are restricted.

More generally, the CR constraints have the form $\alpha \Rightarrow \lambda_1\,\_\,\rho_1, ..., \lambda_k\,\_\,\rho_k$, where k is the *number of contexts*, the variable α stands for the pattern, and the variables $\lambda_1, ..., \lambda_k, \rho_1, ..., \rho_k$ constitute k contexts in pairs. Each variable is a recognizable subset of Σ^* and is usually given through a regular expression. We assume that every context ($\lambda \_ \rho$) is *total* i.e. it reaches the word boundaries although most implementations (e.g. FOMA) require word boundary markers (**.#.** $\lambda \_ \rho$ **.#.**) in this case.

In the original use scenario – Two Level Morphology – CRs typically restrict allophonemes or allomorphophonemes to a relatively small number of contexts. More recently, CR has been used to express complete morphological lexicons in which case the total number of contexts can be in thousands (Yli-Jyrä, 2009).

CR constraints can also be applied to morphosyntax and surface syntax where the total number of conjunctive CR constraints can be significantly higher than in phonology. In an early manifestation of Finite State Intersection Grammar (FSIG) (Koskenniemi et al., 1992; Yli-Jyrä, 2003), both the patterns and the contexts were linguistically informed but quite complicated. More recent FSIG formalisms focus on local bracketed tree constraints (Yli-Jyrä, 2005) that are motivated by the well-known succinctness characteristics of packed forests.

Some of the decision problems for extended regular expressions and CRs in particular (Måns Hulden, pers.comm. 2010) are intractable and the state complexity of a compiled CR can be prohibitively large in the worst case scenarios. Therefore, the basic research on CR compilation algorithms increasingly tries to identify *islands of tractability* for the CR compilation problem so that larger systems could exploit CR and context-sensitive constraints without efficiency bottlenecks.

1.2 Prior Compilation Techniques

There are six important approaches (SF, GR, FO, IC, PF, WL) to the compilation of the CR constraints. Let us describe each of them in turn.

1. **SF: Star-Free Regular Expressions**

   The closure of the empty set $\emptyset$ and the singleton languages $\{w\}$ ($w \in \Sigma^*$) under the operations of concatenation ($\cdot$), intersection ($\cap$) and complement ($^-$) defines the *star-free* i.e. *counter-free* languages (McNaughton and Papert, 1971). Note that $\overline{\alpha \cap \beta} = \overline{\alpha} \cup \overline{\beta}$ (by De Morgan's laws) and $\overline{\emptyset} = \Sigma^*$. The star-free operators ($\cdot$, $\cap$, $\cup$, $^-$) give a compilation formula for CRs with $k = 1$ (Koskenniemi, 1983, 106):

$$\overline{\overline{\lambda_1}\alpha\overline{\emptyset}} \cap \overline{\overline{\emptyset}\alpha\overline{\rho_1}} \tag{2}$$

   However, the length of the formula grows exponentially when k grows, being already high for two bilateral contexts (Yli-Jyrä, 2003):

$$\overline{\overline{\lambda_2}\alpha\overline{\rho_1}} \cap \overline{\overline{\lambda_1}\alpha\overline{\rho_2}} \cap \overline{(\overline{\lambda_1} \cap \overline{\lambda_2})\alpha\overline{\emptyset}} \cap \overline{\overline{\emptyset}\alpha(\overline{\rho_1} \cap \overline{\rho_2})}. \tag{3}$$

2. **IC: Indexed Contexts**

   Karttunen et al. (1987) observe that when the pattern language α consists of atomic symbols, a multi context CR can be decomposed into simple CRs. This is done by indexing every atomic symbol in α by the context pairs $\lambda_i \_ \rho_i$. The idea has two implementations:

   (a) In Karttunen et al. (1987) and Kaplan and Kay (1994), the atomic symbols are surrounded by indexed brackets. The method needs an extended alphabet such as $\Sigma \cup \{\langle_1, ..., \langle_k\} \cup \{\rangle_1, ..., \rangle_k\}$.

   (b) In Koskenniemi and Silfverberg (2010), the atomic symbols are themselves indexed. This method needs an extended alphabet such as $\Sigma \times \{1, ..., k\}$. Thus, each symbol in α is divided into k different variants, $\alpha_1, ..., \alpha_k$.

   In order to reflect the extended alphabet Σ', languages $\lambda_1, ..., \lambda_k, \rho_1, ..., \rho_k$ have to be replaced with the expanded ones $\lambda'_1, ..., \lambda'_k, \rho'_1, ..., \rho'_k$. The indexing is later cancelled by an appropriate homomorphism $g : \Sigma'^* \to \Sigma^*$ and the CR semantics is finally given by the formula

$$g(\cap_{i=1}^{k} \alpha_i \Rightarrow \lambda'_i \_ \rho'_i). \tag{4}$$

   The recognizer for the result is nondeterministic, but it is unclear if the result is ever smaller than a canonical automaton. Nevertheless, the extended alphabet has an undesirable effect on the number of transitions in various stages of the compilation process.

   The approach has been generalized beyond the atomic symbols to a slightly larger family of patterns languages: a modified IC formula applies to the case where, for all $w, w' \in \alpha$, the words w and w' are overlap-free or equivalent (Yli-Jyrä and Koskenniemi, 2004, formula 16 on page 18). The formula resembles an implementation of replace rules Kempe and Karttunen (1996) and is related to a former implementation of the CR operator in Xerox Finite State Tool (XFST) (Yli-Jyrä, 2003; Yli-Jyrä and Koskenniemi, 2004; Karttunen, 2004).

3. **GR: SF with a Restricted Homomorphism**

   The *generalized restriction (GR)* operation (Yli-Jyrä and Koskenniemi, 2004) increases the compactness of star-free expressions by using a marker alphabet Δ, such that $\Delta \cap \Sigma = \emptyset$, and an operation that removes a finite number of markers from words.

   Let $\Sigma_\Delta = \Sigma \cup \{\Delta\}$ and let $f : \Sigma_\Delta^* \to \Sigma^*$ be a string homomorphism defined by $h(a) = a$, $h(\diamond) = \epsilon$, $h(\epsilon) = \epsilon$, $h(x \cdot y) = h(x) \cdot h(y)$ for all $a \in \Sigma$, $\diamond \in \Delta$. Within the method, the homomorphism h can be replaced with its restriction $h_d = h_{|(\overline{\emptyset}(\Delta\overline{\emptyset})^d)}$ where $d \in \mathbb{N}$. Since star-free languages are closed under the restriction of h_d, it extends star-free regular expressions.

   For all $d \in \mathbb{N}$, the GR operation is syntactically represented by the operator $\overset{d\Delta}{\Longrightarrow}$ whose semantics is defined over d-marker languages $W, W' \subseteq \overline{\emptyset}(\Delta\overline{\emptyset})^d$ by

   $$W \overset{d\Delta}{\Longrightarrow} W' = \overline{h_d(W - W')}. \tag{5}$$

   We will call the left side, W, the *pattern (language)* and the opposite side, W', the *context (language)*. Let $\Delta = \{\diamond\}$. The semantics of a CR constraint is expressed by:

   $$(\overline{\emptyset}\diamond\alpha\diamond\overline{\emptyset}) \overset{2\Delta}{\Longrightarrow} \cup_{i=1}^{k}(\lambda_n\diamond\alpha\diamond\rho_n). \tag{6}$$

   Note that (5) requires a deterministic or co-deterministic recognizer for W'.

4. **FO: A Fragment of Second-Order Logic**

   Various logical formalisms have been used for defining the semantics of the CR operation exactly (Koskenniemi, 1983; Yli-Jyrä and Koskenniemi, 2004; Vaillette, 2004; Hulden, 2008). Koskenniemi (1983, 36) defines a CR with $k = 1$ through a logical expression (7) but did not explicate any model-theoretic semantics of the logic itself.

   $$\begin{aligned}&\{w|(w = vxy \wedge x \in \alpha)\\ &\to (v \in \Sigma^*\lambda_1 \wedge y \in \rho_1\Sigma^*)\}.\end{aligned} \tag{7}$$

   In finite model theory, the semantics of the CR operation can be defined precisely through monadic second-order logic (MSO) or, if the operands are star-free, in its first-order (FO) fragment. In both cases, the formula is interpreted over finite words. The semantics of MSO relies on automata over an extended alphabet, reflecting the power set of the variables in the formula. For example, variables v and y would induce the extended alphabet $\Sigma' = \Sigma \times 2^{\{v,y\}}$.

   In Yli-Jyrä and Koskenniemi (2004) and Hulden (2008), the semantics of FO variables has been defined with markers. The markers are often cheaper than the set-based encoding of FO variables as they extend the alphabet only by one new symbol per variable.

   Customized predicate logics (Vaillette, 2004; Hulden, 2008) add syntactic sugar to the usual MSO logic through substring variables $x, y, z, ...$. In addition, regular expressions α, λ_1, ρ_1 etc. can be used in the predicates. With these extensions, the model theoretic semantics of CR can be expressed through such elegant formulas as

   $$(\forall x)(\textit{matches}(x, \alpha) \to \textit{btw}(x, \lambda_1, \rho_1)). \tag{8}$$

5. **PF: Prefix-Free Patterns**

   If we assume that that the pattern α (as a set of strings) does not contain proper prefixes (symmetrically: proper suffixes), one marker in the GR pattern language becomes redundant. This observation helps to reduce the 2-marker GR approach to a 1-marker GR approach whenever the assumption holds for the patttern α:

   $$(\overline{\emptyset}\diamond\alpha\overline{\emptyset}) \overset{1\Delta}{\Longrightarrow} \cup_{i=1}^{k}(\lambda_n\diamond\alpha\rho_n).$$

CRs with prefix-free patterns occur naturally inside the XFST-style replace rules (Yli-Jyrä, 2008b), bracketed FSIGs (Yli-Jyrä, 2008a), and partition-based grammars (Grimley-Evans et al., 1996). In these applications, each pattern match can be unambiguously marked with one marker only.

The assumption of prefix-freeness is trivially true when $\alpha \subseteq \Sigma$. This assumption was used in Yli-Jyrä (2009), where also an optimized compilation algorithm for the 1-marker GR was presented.

A variant of this approach is in place in Partition Based Two Level Morphology (Grimley-Evans et al., 1996) where multi character patterns form adjacent blocks into which the whole word is implicitly partitioned. As each of the block is bracketed, the blocks cannot be prefixes or suffixes of one another.

6. **WL: Weighted Logic**

   The weighted MSO logic (Droste and Gastin, 2009) can be used to define the characteristic series $\mathbb{1}_L \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ of any recognizable language $L \in \Sigma^*$ and, in particular, of the language of the constraint $\alpha \Rightarrow \lambda_1 \_ \rho_1, ..., \lambda_k \_ \rho_k$. For any word $w \in c_1...c_n \in \Sigma^*$, assume that $\alpha'(v,y), \lambda'_1(v), ..., \lambda'_k(v), \rho'_1(y), ..., \rho'_k(y)$ are appropriate wMSO$(\mathbb{B},\Sigma)$ formulas defining the membership tests $(c_v...c_y \in \alpha)$, $(c_1...c_{v-1} \in \lambda_1)$, ..., $(c_1...c_{v-1} \in \lambda_k)$, $(c_{y+1}...c_n \in \lambda_1)$, ..., $(c_{y+1}...c_n \in \lambda_k)$. Then the formula

   $$\forall v.\forall y.(\alpha'(v,y) \rightarrow \vee_{i=1}^{k}\lambda'_i(v) \wedge \rho'_i(y))$$

   defines the characteristic series $\mathbb{1}_L$ for the constraint $\alpha \Rightarrow \lambda_1 \_ \rho_1, ..., \lambda_k \_ \rho_k$.

   Each universally quantified variable must be eliminated separately because each elimination asserts that the quantifier's scope is expressing a *recognizable step function* (Droste and Gastin, 2009). This is contrasted to the (unweighted) predicate logic of Hulden (2008) where the quantified variables are defined over position pairs. The elimination of the weighted universal quantifiers has been described in the proof of Lemma 5.4 in Droste and Gastin (2009)

In sum, the prior CR compilation methods can be characterized, on average, by the following properties:

1. a product alphabet (IC, (FO,) WL)
2. the pattern-context contrast (all)
3. substring quantification (all but PF, WL)
4. relies on deterministic automata (all)
5. uses De Morgan duals (nearly all)
6. unavoidable DFA result (SF, GR, FO, PF).

1.3 The Overview of the Unconventional Approach

The method presented in the following sections is nonconventional in many respects as it takes advantage of the following observations:

1. **O(1) Markers.** The GR method has demonstrated that adding $O(1)$ markers to the alphabet is enough. We will thus use markers instead of a heavily extended alphabet. This also means that we start our thinking from the GR operation.

2. **Patternless GR.** One of the operands of the GR operation can be eliminated as the pattern $W \subseteq \overline{\emptyset}\diamond\overline{\emptyset}$ can be moved to the right side of the GR without any effect on the semantics: $\left(W \overset{1\Delta}{\Longrightarrow} W'\right) = \left(\overline{\emptyset}\diamond\overline{\emptyset} \overset{1\Delta}{\Longrightarrow} (W' \cup \overline{W})\right)$. The resulting *patternless* GR can be viewed as a form of a universal quantifier:

   $$\left(\overline{\emptyset}\diamond\overline{\emptyset} \overset{1\Delta}{\Longrightarrow} W''\right) = \{c_1...c_n \mid \forall i\in\{0,...,n\}.c_1...c_i\diamond c_{i+1}...c_n \in W''\}. \quad (9)$$

3. **One Position.** The quantified positions can be eliminated one by one. A patternless 2-marker GR where $\Delta = \{\diamond_1, \diamond_2\}$ and $W' \subseteq W = \overline{\emptyset}\diamond_1\overline{\emptyset}\diamond_2\overline{\emptyset}$ reduces to a pair of nested 1-marker GRs:

   $$\left(W \overset{2\Delta}{\Longrightarrow} W'\right) = \left(\overline{\emptyset}\diamond\overline{\emptyset} \overset{1\{\diamond_1\}}{\Longrightarrow} \left(\overline{\emptyset}\diamond\overline{\emptyset} \overset{1\{\diamond_2\}}{\Longrightarrow} W'\right)\right). \quad (10)$$

The approach is comparable to weighted logic where one *cannot generally* remove two universally quantified variables at once because the resulting weighted automaton is not necessarily finite.

4. **Determinism and Co-determinism.** Combinations of left and right sequential transducers (Johnson, 1972; Skut et al., 2004; Peikov, 2006) have been applied in the compilation of context-dependent rewriting rules. Analogously, the recognizer for the context language W' can be determinized and co-determinized "inwards", towards the marker (Section 2.3).

5. **No Complementations.** The local structure of the "inward" deterministic recognizer for the pattern language W' can be used directly as if it were a readily compiled constraint (Section 3). Thus, the "double complementation" needed by many prior methods is avoided.

6. **NFA Result.** The compilation can result into a nondeterministic automaton (NFA). In applications, NFAs can be used as constraints since they are closed under the intersection operation.

The rest of the paper is committed to the realization of the new core operation: the patternless GR $(\overline{\emptyset}\diamond\overline{\emptyset} \overset{1\{\diamond_2\}}{\Longrightarrow} W')$. A patternless GR operation can express a CR or even a combination of CRs. This operation is described in Section 3.

Before the section, some prerequisites are given (Section 2), and after the section, the paper is concluded with complexity analysis and remarks (Section 4).

2 The Prerequisites

2.1 Automata

For overviews and the terminology of finite automata, the reader is referred to a text book such as Hopcroft et al. (2006).

A (nondeterministic) *finite automaton* (fa) is a 5-tuple $\mathcal{A}{=}(Q, I, F, \Sigma, \delta)$ with states Q, initial states I, final states F, input alphabet Σ and the transition relation $\delta \subseteq Q \times \Sigma \times Q$. For every state $q \in Q$ and letter $a \in \Sigma$, the set of states $\{r | (q, a, r) \in \delta\}$ reached by input a is denoted by qa. Extend the notation to a state set $P \subseteq Q$ and a word $w = a_1...a_n \in \Sigma^*$ in such a way that $Pa = \{q \mid p \in P, (p, a, q) \in \delta\}$ and $Pa_1...a_2 = (Pa_1)a_2...a_n$. The automaton recognizes the language $||\mathcal{A}|| = \{w \in \Sigma^* \mid Iw \cap P \neq \emptyset\}$.

Let $\mathcal{A} = (Q, I, F, \Sigma, \delta)$ be a fa. Denote its structural reversal $(Q, F, I, \Sigma, \delta')$ where $\delta' = \{(r, a, q) \mid (q, a, r) \in \delta\}$ by $\mathcal{A}^{\mathrm{R}}$. Denote by $\Sigma_\Delta = \Sigma \cup \Delta$ an alphabet such that $\Delta \cap \Sigma = \emptyset$.

Algorithm 1 BARRIERDET$(\mathcal{A}, \Delta)$: Determinization until Δ-barrier

```
Require: A fa $\mathcal{A} = (Q, I, F, \Sigma_\Delta, \delta)$
 1: $done \leftarrow F' \leftarrow \delta' \leftarrow \emptyset$; $Q' \leftarrow \{(0, I)\}$
 2: while $Q' \neq done$ do
 3:   Pick a state $(s, P)$ from $Q' - done$;
      Insert the state $(s, P)$ to $done$
 4:   for all $a \in \Sigma_\Delta$ with $Pa \neq \emptyset$ do
 5:     if $s = 0$ and $a \in \Sigma$ then
 6:       Insert the state $(0, Pa)$ to $Q'$;
          Insert the triplet $((0, P), a, (0, Pa))$ to $\delta'$
 7:     else
 8:       for all $r \in Pa$ do
 9:         insert $((s, P), a, (1, \{r\}))$ to $\delta'$, and
            $(1, \{r\})$ to $Q'$
10:       end for
11:     end if
12:   end for
13: end while
14: return $\mathcal{A}'{=}(Q', \{(0, I)\}, Q'{\cap}(\{1\}{\times}F), \Sigma_\Delta, \delta')$
Ensure: (see the referrence in the text)
```

2.2 Barrier Deterministic Automata

Definition 2.1. *Let* $\mathcal{A} = (Q, I, F, \Sigma_\Delta, \delta)$. *The fa* $\mathcal{A}$ *is* barrier deterministic with respect to the marker set Δ *if*

1. *there is at most one initial state, i.e.* $|I| \leq 1$

2. *the states* Q *can be divided into the sets of left and right states* Q_1 *and* Q_2 *in such a way that* $\delta \subseteq (Q_1 \times \Delta \times Q_2) \cup (Q_1 \times \Sigma \times Q_1) \cup (Q_2 \times \Sigma \times Q_2)$

3. *the left states are deterministic i.e.* $|qa| \leq 1$ *for every state* $q \in Q_1$ *and letter* $a \in \Sigma$.

Let $\mathcal{A}=(Q, I, F, \Sigma_\Delta, \delta)$ be a fa. Algorithm 1 implements a function called BARRIERDET. BARRIERDET$(\mathcal{A}, \Delta)$ is a barrier deterministic automaton $\mathcal{A}' = (Q_1 \cup Q_2, I', F', \Sigma_\Delta, \delta')$ with

- $|I| \leq 1$, $Q_1 \cap Q_2 = \emptyset$, $\delta \subseteq (Q_1 \times \Delta \times Q_2) \cup (Q_1 \times \Sigma \times Q_1) \cup (Q_2 \times \Sigma \times Q_2)$, and $|qa| \leq 1$ for every state $q \in Q_1$ and letter $a \in \Sigma$.
- $Q_2 \subseteq \{1\} \times Q$ and $\delta' \cap (Q_2 \times \Sigma \times Q_2) = (\{1\} \times \delta) \cap (Q_2 \times \Sigma \times Q_2)$
- $||\mathcal{A}'|| = ||\mathcal{A}|| \cap \Sigma^* \Delta \Sigma^*$.

2.3 Inward Deterministic Automata

Definition 2.2. *Let $\mathcal{A} = (Q, I, F, \Sigma_\Delta, \delta)$. The automaton $\mathcal{A}$ is* inward deterministic with respect to the marker set Δ *if both $\mathcal{A}$ and $\mathcal{A}^{\mathrm{R}}$ are barrier deterministic with respect to the marker set Δ.*

An inward deterministic automaton $\mathcal{A}'' =$ INWARDDET$(\mathcal{A}, M)$ with $||\mathcal{A}''|| = ||\mathcal{A}|| \cap \Sigma^* \Delta \Sigma^*$ is given by

$$\text{INWARDDET}(\mathcal{A}, M) = \text{BARRIERDET}(\text{BARRIERDET}(\mathcal{A}, \Delta)^{\mathrm{R}}, \Delta)^{\mathrm{R}}. \quad (11)$$

Note that $\mathcal{A}''$ has at most one path for every $v \diamond y \in \Sigma^* \Delta \Sigma^*$ where $\diamond \in \Delta$. The definition is illustrated in Figure 1.

3 Compiling Patternless GR Constraints

Let $\mathcal{A} = (Q, I, F, \Sigma_{\{\diamond\}}, \delta)$ be an automaton that is inward deterministic with respect to $\{\diamond\}$. This section describes how a recognizer for the language $\Sigma^* - h_1(\Sigma^* \diamond \Sigma^* - ||\mathcal{A}||)$ of the patternless GR constraint $(\overline{\emptyset} \diamond \overline{\emptyset} \overset{1\{\diamond\}}{\Longrightarrow} W')$ with $W' = ||\mathcal{A}||$ is constructed.

The language $L = \Sigma^* - h_1(\Sigma^* \diamond \Sigma^* - W')$ is described either through a double complement or positively:

- Any word $w \in \Sigma^*$ such that $h_1^{-1}(w) \not\subseteq W'$ is "nogood" i.e. $w \notin L$
- Any word $w \in \Sigma^*$ such that $h_1^{-1}(w) \subseteq W'$ is "good" i.e. $w \in L$.

The positive description for L has an interpretation in an inward deterministic recognizer $\mathcal{A}$. Let $w = c_1 ... c_n \in \Sigma^*$. If $w \in L$ then $\mathcal{A}$ has two disjoint w-labeled (incomplete) paths:

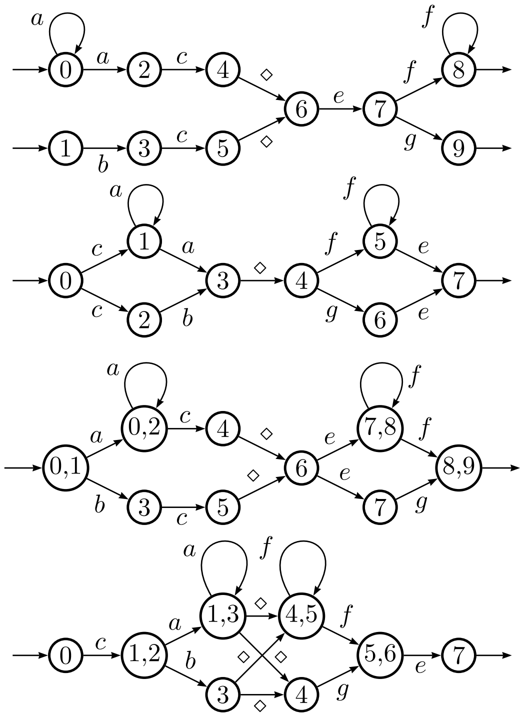

Figure 1: The lower two automata are inward deterministic (with $\Delta = \{\diamond\}$), while the upper two automata are not inward deterministic.

1. An initial left context path $\pi_l = \langle l_0, l_1, ..., l_n \rangle$ that starts from the initial state $l_0 = i$ and ends at the state l_n.
2. A final right context path $\pi_r = \langle r_0, r_1, ..., r_n \rangle$ that starts from the final state $r_n = f$ and proceeds left-deterministically to the state r_0.

These two paths in the automaton $\mathcal{A}$ are connected with $n+1$ transitions on the $\diamond$-marker, and they thus form $n+1$ complete runs, one for each $n+1$ marked word in the language $h_1^{-1}(w)$. They constitute a ladder-shaped substructure as illustrated in Figure 2.

Let $Z \subseteq \Sigma^* \diamond \Sigma^*$ be a marked language. Then Z is *closed* under variant markings, if $Z = h_1^{-1}(h_1(Z))$. In this sense, the largest closed subset of $||\mathcal{A}||$ is LADDER$(||\mathcal{A}||) = \Sigma^* \diamond \Sigma^* - h_1^{-1}(h_1(\Sigma^* \diamond \Sigma^* - ||\mathcal{A}||))$. We now have the equivalence between the double complement description and the positive description for the language L:

$$\Sigma^* - h_1(\Sigma^* \diamond \Sigma^* - W') = h_1(\text{LADDER}(||\mathcal{A}||)).$$

SUPERPOSE$(\mathcal{A}, \diamond)$ (Algorithm 2) detects the sublanguage LADDER$(||\mathcal{A}||)$ from $\mathcal{A}$ and constructs

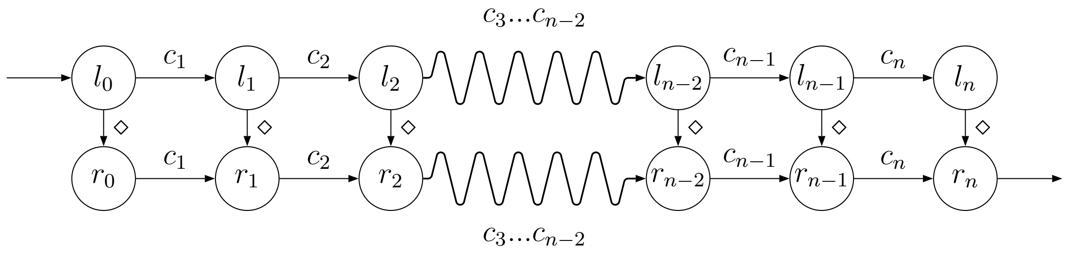

Figure 2: The ladder-shaped substructure of $\mathcal{A}$ corresponding to the word $c_1 \ldots c_n$.

a recognizer $\mathcal{A}''$ for its homomorphic image $h_1(\text{LADDER}(||\mathcal{A}||))$. The lines 3-6 optimize the algorithm by restricting its state set to the accessible part. In practice, this optimization can be easily incorporated to the main construction that is on lines 1 and 2.

Algorithm 2 SUPERPOSE($\mathcal{A}$,$\diamond$)

```
Require: Inward deterministic NFA
      $\mathcal{A} = (Q, \{i\}, \{f\}, \delta)$ with $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*\diamond\Sigma^*$
1: $Q' \leftarrow \{(l,r) \mid (l,\diamond,r) \in \delta\}$;
   $I' \leftarrow \{(l,r) \in Q' \mid l \in I\}$;
   $F' \leftarrow \{(l,r) \in Q' \mid r \in F\}$
2: $\delta' \leftarrow \{((l,r),a,(l',r')) \mid$
         $(l,r),(l',r') \in Q', l' \in la, r' \in ra\}$
3: $\mathcal{A}' = (Q', I', F', \Sigma, \delta')$; $P \leftarrow I'$
4: while $P\Sigma - P \neq \emptyset$ do
5:    $P \leftarrow P \cup P\Sigma$
6: end while
7: return $\mathcal{A}''=(P, I', F' \cap P, \Sigma, \delta'(\cap P \times \Sigma \times P))$
Ensure: $||\mathcal{A}''|| = h_1(\text{LADDER}(||\mathcal{A}||))$
```

4 Evaluation

4.1 The Worst-Case Complexity Analysis

The complexity of INWARDDET (Algorithm 1) alone is similar to the general determinization algorithms: exponential to the size of the input. Let l and r be the size of the left-context component and the right-context component of the automaton that is an input to the inward determinization algorithm. Denote the left-context and right-context state sets of the result of INWARDDET, respectively, by Q and R. The respective sizes of the Q and R components of the inward deterministic result are then $O(2^l)$ and $O(2^r)$.

The SUPERPOSE algorithm (Algorithm 2) assumes a nondeterministic automaton that is inward deterministic with respect to the marker $\diamond$. Such an automaton contains $O(|Q||R|)$ marker transitions and $O((|Q| + |R|)|\Sigma|)$ normal transitions because the states have only one transition per a letter. The size of the projection of the inward deterministic automaton is polynomial to the size of the inward deterministic automaton. Namely, it contains $O(|Q||R|)$ states that corresponds to $\diamond$-transitions in the input. Since the result is nondeterministic, a state (q, r) can have, for every letter $a \in \Sigma$, a transition to any of the states $\{(q', r')|\delta(q, a) = \{q'\}, r' \in \delta(r, a)\}$. The total number of transitions is $O(|Q||R|^2|\Sigma|)$. Based on this, the time complexity of the SUPERPOSE is $O(|Q||R|^2|\Sigma|)$ if we assume that each of result transitions can be created in constant time.

The worst case nondeterministic state complexity of the output of the INWARDDET and SUPERPOSE methods is $O(2^l(2^r)^2|\Sigma|)$ i.e. $O(2^s)$ where $s = l + r$. Recall that this is applicable to patternless GRs only. The patternless GR is directly usable when the pattern language α of the CR constraint is prefix-free or suffix-free. In all other cases, the compilation of CRs requires more general, but less efficient methods that involve two markers (possibly through nested patternless GRs).

4.2 Updating the Best Practice

The comparative sizes of minimal deterministic, codeterministic and inward deterministic representations of the context language W' may differ significantly. For example, the deterministic or codeterministic automaton recognizing the language $\Sigma^n a\Sigma^*\diamond\Sigma^* a\Sigma^n$ (for any large enough $n \in \mathbb{N}$) is exponentially larger than the corresponding inward deterministic automaton. The comparative size difference means that using (INWARDDET+)SUPERPOSE to compile this patternless GR would be an es-

smallest representation		recommendation	
for λ	for ρ	for $\lambda\diamond\rho$	method
determ.	determ.	determ.	GR with DFAs
determ.	co-det.	inw.det.	SUPERPOSE
co-det.	co-det.	co-det.	GR with r-DFAs

Table 1: The choice for the compilation method when other known methods cause immediate blow-up in the representation of contexts.

sential improvement over the state-of-the-art methods where determinization (GR with DFAs) or co-determinization (GR with reverse DFAs) is the first step of the compilation. There are also opposite situations where (INWARDDET+)SUPERPOSE is less likely the most appropriate method (Table 1). Clearly, this kind of rules of thumb will be refined when we can evaluate the predictions with additional practical experiments.

The differences between the efficiency of the methods are often less dramatic in practice. The initial experiments with some 1100 CR constraints from a syntactic FSIG grammar (Voutilainen, 1997) indicate that the size of the inward deterministic automaton is typically very close (1.0 - 4.0 $\times$) to the corresponding minimal deterministic automaton. More careful implementation and experiments are needed in order to find significant differences in efficiency.

According to the author's experiences, it is complicated to add the barrier and inward determinization algorithms to the existing finite-state libraries and tools. Namely, many finite state tools, such as XFST, FOMA, and SFST, typically store the intermediate results as canonical automata. Therefore, the current work suggests that the tools should handle also nondeterministic and co-deterministic automata as full citizens of the finite-state calculus.

Perhaps the cleanest way to add the currently presented algorithms to finite state libraries is to encapsulate the barrier determinization, the reversal and the SUPERPOSE algorithm into one routine where they can store and optimize the nondeterministic automata as needed. However, this seems to be counterproductive from the perspective of reusability.

Perhaps the best practice for using the currently presented method is to use multiple methods and avoid expensive determinizations whenever possible.

4.3 Further Work

There are some possibilities for optimizations and extensions in the presented algorithms: (1) The inward determinization can be optimized by adding some filtering for states that cannot be used by the SUPERPOSE algorithm. (2) A notion of minimality can be adapted to inward deterministic automata and the minimized inward deterministic automata can help reduce the size of the compiled result. (3) The current method for CR compilation could be embedded in methods that compile replace rules or methods where the constraints or rules are compiled on "the fly". (4) a weighted variant of the current method should be compared against Droste and Gastin (2009).

Acknowledgments

This work was supported by the Academy of Finland grant number 128536 "Open and Language Independent Automata-Based Resource Production Methods for Common Language Research Infrastructure". I am also indebted to Måns Hulden, Andreas Maletti and Jacques Sakarovitch and the anonymous reviewers for their critical and constructive remarks.

References

Manfred Droste and Paul Gastin. 2009. Weighted automata and weighted logics. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science, chapter 5, pages 175–211. Springer-Verlag, Berlin Heidelberg.

Edmund Grimley-Evans, George Anton Kiraz, and Stephen G. Pulman. 1996. Compiling a partition-based two-level formalism. In *16th International Conference on Computational Linguistics (COLING 1996)*, volume 1, pages 454–459, Center for Sprogteknologi, Copenhagen, Denmark.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Måns Hulden. 2008. Regular expressions and predicate logic in finite-state processing. In Jakub Piskorski, Bruce Watson, and Anssi Yli-Jyrä, editors, *Finite-state methods and natural language processing*. IOS Press, Amsterdam, The Netherlands.

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.

Lauri Karttunen, 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.

Lauri Karttunen. 2004. Restriction operator error: Technical note. Web page `http://www.stanford.edu/~laurik/fsmbook/errata/restriction-operator.html` read on Oct 7, 2011.

André Kempe and Lauri Karttunen. 1996. Parallel replacement in finite state calculus. In *16th International Conference on Computational Linguistics (COLING 1996)*, volume 2, pages 622–627, Center for Sprogteknologi, Copenhagen, Denmark.

Kimmo Koskenniemi and Miikka Silfverberg. 2010. A method for compiling two-level rules with multiple contexts. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, SIGMORPHON '10, pages 38–45, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kimmo Koskenniemi, Pasi Tapanainen, and Atro Voutilainen. 1992. Compiling and using finite-state syntactic rules. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING'92*, volume 1, pages 156–162. International Committee on Computational Linguistics, Nantes, France.

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

Robert McNaughton and Seymour Papert. 1971. *Counter-Free Automata*. Number 65 in Research Monograph. MIT Press, Cambridge, Massachusetts.

Ivan Petrov Peikov. 2006. Direct construction of a bimachine for context-sensitive rewrite rule. Master's thesis, Sofia University St. Kliment Ohridski, Faculty of Mathematics and Computer Science, Department of Mathematical Logic and Applications, Sofia, Bulgaria.

Wojciech Skut, Stefan Ulrich, and Kathrine Hammervold. 2004. A bimachine compiler for ranked tagging rules. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA.

Nathan Vaillette. 2004. *Logical Specification of Finite-State Transductions for Natural Language Processing*. Ph.D. thesis, Department of Linguistics, Ohio State University.

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

Anssi Yli-Jyrä and Kimmo Koskenniemi. 2004. Compiling contextual restrictions on strings into finite-state automata. In *The Eindhoven FASTAR Days, Proceedings*, number 04/40 in Computer Science Reports, The Netherlands. Technische Universiteit Eindhoven.

Anssi Yli-Jyrä. 2003. Describing syntax with star-free regular expressions. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 379–386, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anssi Yli-Jyrä. 2005. *Contributions to the Theory of Finite-State Based Grammars*. Number 38 in Publications. Department of General Linguistics, University of Helsinki.

Anssi Yli-Jyrä. 2008a. Applications of diamonded double negation. In Thomas Hanneforth and Kay-Michael Würzner, editors, *Finite-State Methods and Natural Language Proceedings. 6th International Workshop, FSMNLP 2007, Potsdam, Germany, September 14-16. Revised Papers*, pages 6–30. Potsdam University Press, Potsdam, Germany.

Anssi Yli-Jyrä. 2008b. Transducers from parallel replacement rules and modes with generalized lenient composition. In Thomas Hanneforth and Kay-Michael Würzner, editors, *Finite-State Methods and Natural Language Proceedings. 6th International Workshop, FSMNLP 2007, Potsdam, Germany, September 14-16. Revised Papers*, pages 196–212. Potsdam University Press, Potsdam, Germany.

Anssi Yli-Jyrä. 2009. An efficient double complementation algorithm for superposition-based finite-state morphology. In Kristiina Jokinen and Eckhard Bick, editors, *Proceedings of the 17th Nordic Conference of Computational Linguistics, NODALIDA 2009, May 14-16, 2009*, volume 4 of *NEALT Proceedings Series*, Odense, Danmark. Northern European Association for Language Technology.