

Transducers from Parallel Replace Rules and Modes with Generalized Lenient Composition

Anssi Yli-Jyrä

Department of General Linguistics, University of Helsinki, Finland

Abstract. Generalized Two-Level Grammar (GTWOL) provides a new method for compilation of parallel replacement rules into transducers. The current paper identifies the role of *generalized lenient composition* (GLC) in this method. Thanks to the GLC operation, the compilation method becomes bipartite and easily extendible to capture various application modes. In the light of *three notions of obligatoriness*, a modification to the compilation method is proposed. We argue that the bipartite design makes implementation of parallel obligatoriness, directionality, length and rank based *application modes* extremely easy, which is the main result of the paper.

1 Introduction

It is extremely difficult to compile grammars into finite-state transducers without efficient and readily implemented compilation methods for high-level rules. In particular, replace rules (such as [1]) have a rich semantics that is difficult to capture. The goal of this paper is to analyze the author's recently proposed method [2] and the related approach in general.¹

The new method [2] differs from the most similar alternative approach of Kempe and Karttunen [1] in some obvious ways:

- It reduces oriented replace rules to two-level rules
- It does not necessarily use composition
- It derives all modes from optional replacement
- Its left-and-right context conditions are closed under Boolean operations
- It uses brackets only to avoid overlapping rule applications.

In this paper, perhaps the most important contribution is the recognition of the relevance of the *bipartite architecture* of the new method. According to it, the rule-independent mode constraints are separated from rule-specific condition. Related to this, we present the necessary machinery including Jäger's composition operator [3] and new strict preference relations. The second important contribution is to present *Bracketed Generalized Two-Level Grammar* (BGTWOL) that is crucial to the new compilation method. The third contribution is to separate *three modes of obligatoriness*. A clear understanding of these

¹ For further resources <http://www.ling.helsinki.fi/users/aylijyra/replace>.

modes helps relate the existing compilation methods and improve the compatibility of the new method and the Xerox calculus. Finally, the paper sketches a *rich rule system* that covers the multi-character two-level rules of GTWOL [4,5] and BGTWOL, parallel replace and marking rules [1,6], directed modes [7] and three principles for ranking [8] or disjunctive ordering [5].

The paper is structured as follows: Preliminary definitions are in Section 2. In Section 3, we describe the essentials of Generalized Two-Level Grammar (GTWOL) [4]. Section 4 reduces replace operations into the GTWOL formalism. Section 5 studies applications of generalized lenient composition to obligatory replacement. The new design pattern for compilation methods is discussed in Section 6. The conclusion is in Section 7.

2 Preliminaries

Let A_1, A_2 be sets of symbols. Let U and V be languages over A_1 . We assume that the reader is familiar with regular languages and the basic regular operations: concatenation UV , intersection $U \cap V$, union $U \cup V$, asymmetric difference $U \setminus V$, complementation \bar{U} , Kleene's star U^* , and Kleene's plus U^+ . Let $U^0 = \epsilon$ and let U^k , where $k > 0$, denote the languages $UU^{(k-1)}$.

The *local A_2 -closure* of $U \subseteq A_1^*$ is the relation $f_{A_2}: A_1^* \rightarrow A_1^*$ defined as $f_{A_2}(U) = \{f(a_0)f(a_1)\dots f(a_{m-1}) \mid a_0a_1\dots a_{m-1} \in U \wedge a_0, a_1, \dots, a_{m-1} \in A_1\}$ where $f(a) = a^*$ for every $a \in A_2$, and $f(a) = a$ otherwise. The *elimination* of symbols A_2 in language U is the function $d_{A_2}(U) = f_{A_2}(U) \setminus A_1^*A_2A_1^*$. The inverse of relation d_{A_2} is denoted by $d_{A_2}^{-1}$.

Notation $A_1:A_2$ denotes alphabet $\{a_1:a_2 \mid a_1 \in A_1 \wedge a_2 \in A_2\}$. Set Π is called the *total pivot alphabet*. Its every element is a character pair $a:b$ and it is closed in such a way that $a:a, b:b \in \Pi$ for all $a:b \in \Pi$. The *diamond alphabet* M contains markers $\#:\#, _:_, \diamond_0:\diamond_0, \diamond_1:\diamond_1, \diamond_2:\diamond_2, \dots, \diamond_s:\diamond_s$ and it is disjoint from Π . The indices of the diamonds will be used to indicate the disjunctive ordering level of GTWOL rules. Level 1 is the level of the least specific rules. An identity pair $a:a \in (\Pi \cup M)$ is often written simply as a .

We use marker $_ \in M$ to represent the place for centers in an environment string. The *center extension* with $V \subseteq A_1^*$ is the relation $\sigma_V: (A_1 \cup \{_\})^* \rightarrow A_1^*$ defined as $\sigma_V(U) = \{\sigma(a_0)\sigma(a_1)\dots\sigma(a_{m-1}) \mid a_0a_1\dots a_{m-1} \in U \wedge a_0, a_1, \dots, a_{m-1} \in A_1 \cup \{_\}\}$ where $\sigma(a) = V$ when $a = _$, and $f(a) = a$ otherwise.

The null string is denoted by ϵ . Let u be a string over an alphabet A_1 . We often denote set $\{u\}$ by u . The length of u is denoted by $|u|$. A sequence $u = a_0:b_0a_1:b_1\dots a_{m-1}:b_{m-1} \subseteq (A_1:A_2)^*$ is called a *symbol-pair string* and analyzed alternatively as a string pair $(x_1, x_2) = (a_0a_1\dots a_{m-1}, b_0b_1\dots b_{m-1})$. Pair (x_1, x_2) can be denoted by $x_1:x_2$ when $|x_1| = |x_2|$. String x_1 is called the *input string* and x_2 is called the *output string*.

Disjoint sets $B_L \subseteq \Pi$ and $B_R \subseteq \Pi$ have the same cardinality and they are called the *left* and the *right bracket alphabets*, respectively. Set B_L contains symbols $\langle_1, \langle_2, \dots, \langle_s$, and set B_R contains symbols $\rangle_1, \rangle_2, \rangle, \dots, \rangle_s$. Let $B =$

$B_L \cup B_R$ and $B_i = \{<_i, >_i\}$. The indices of the brackets will be used to denote the ranking level of a ranked rule.

Let $0:0 \in \Pi$ be a representative for the empty string ϵ . The *input and output projections* $\pi_1, \pi_2 : \Pi^* \rightarrow \Pi^*$ are defined respectively as $\pi_1(X) = \{d_0(x_1):d_0(x_1) \mid x_1:x_2 \in X\}$ and $\pi_2(X) = \{d_0(x_2):d_0(x_2) \mid x_1:x_2 \in X\}$ where $d_0 = d_{\{0:0\}}$. Let $I = \pi_1(\Pi)$ and $\Sigma = I \setminus B$.

Let $U_2 = \Pi^* M \Pi^* M \Pi^*$. Define relations $\nu_{*,l}, \nu_{2,l} : \Pi^* \rightarrow (\Pi \cup M)^*$ by equations $\nu_{*,l}(w) = d_{\{\circ_1, \circ_2, \dots, \circ_l\}}^{-1}(w)$ and $\nu_{2,l}(w) = \nu_{*,l}(w) \cap U_2$, and relations $\mu, \mu_4 : (\Pi \cup M)^* \rightarrow (\Pi \cup M)^*$ by equations $\mu(w) = \{\#v\nu_{*,l}(x)y\# \mid \circ_j \in M \wedge v, x, y \in \Pi^* \wedge \#v\circ_j x\circ_j y\# \in W\}$ and $\mu_4(w) = \mu(w) \cap \#U_2\#$.

Let $W, W' \in (\Pi \cup M)^*$. The language $\Pi^* \setminus d_M(W \setminus W')$ is denoted by *generalized restriction* $W \xrightarrow{\Pi, \mu_4^M} W'$, if $W \subseteq \#U_2\#$, and by *extended generalized restriction* $W \xrightarrow{\Pi, \mu, M} W'$, if $W = \mu(Y)$ and $W' = \mu(Y')$ for some $Y, Y' \subseteq \#U_2\#$.

It holds that $[W \xrightarrow{\Pi, \mu_4^M} \mu_4(W')] = [W \xrightarrow{\Pi, \mu_4^M} \mu(W')]$ and $[\mu_4(W) \xrightarrow{\Pi, \mu_4^M} \mu(W')] = [\mu(W) \xrightarrow{\Pi, \mu, M} \mu(W')]$. Accordingly, $[\mu_4(W) \xrightarrow{\Pi, \mu_4^M} \mu_4(W')] = [\mu(W) \xrightarrow{\Pi, \mu, M} \mu(W')]$.

3 Generalized Two-Level Grammars

The formalism of Generalized Two-Level Grammars (GTWOL) [4,5] presents several improvements over the classical Two-Level formalism [9,10] in computational morphology. Its main improvement is to support multi-character changes while not turning the formalism into so-called partition-based two-level system which would behave quite differently. Since Yli-Jyrä [5] adds disjunctive ordering to the definition of GTWOL grammars, we will use the same notation here. However, we adopt in this paper an extended notion of the GR operation.

Simple and Complex Rules For any $i \in \mathbb{N}$, let X_i, L_i and R_i denote regular languages over Π , and let l_i be a positive integer. The GTWOL formalism [4,5] includes *center prohibition rule* [$l_i :: X_i / \leq L_i _ R_i$], *context restriction rule* [$l_i :: X_i \Rightarrow L_i _ R_i$], *surface coercion rule* [$l_i :: X_i \leq L_i _ R_i$], and *composite i.e. double-arrow rule* [$l_i :: X_i \Leftrightarrow L_i _ R_i$] that is a short-hand notation for a context restriction rule and a surface rule. The symbols $_$ and $\#$ belong to the diamond alphabet M . Each context condition $C_i = \#L_i _ R_i\# \subseteq \#\Pi^* _ \Pi^*\#$ can be represented by a weaker form $C'_i \subseteq (\epsilon \cup \#)\Pi^* _ \Pi^*(\epsilon \cup \#)$ that is related to C_i by the following equivalence:

$$C_i = [(\epsilon \cup \#\Pi^*)C'_i(\epsilon \cup \#\Pi^*)] \cap (\#\Pi^* _ \Pi^*\#). \quad (1)$$

In other words, the following syntactic conventions are implemented: $\dots_ \Leftrightarrow \dots\epsilon_ \epsilon\dots; \#\Pi^* L_ \dots \Leftrightarrow L_ \dots; \dots_ R \Pi^* \# \Leftrightarrow \dots_ R$. The GTWOL formalism supports rules that have multiple contexts or, more generally, even Boolean combinations of two-sided context conditions, because these context conditions are actually languages.

Let the set of rule operators O contain symbols $/<=, <=, =>, <=>$. The rule types have a general form $X_i \text{ op}_i C_i$, where $X_i \in \Pi^*$, $\text{op}_i \in O$, and $C_i \subseteq \# \Pi^* _ \Pi^* \#$.

The Individual Rules of GTWOL The semantics of the individual rules of GTWOL grammar is defined as follows:

$$[l_i :: X_i / <= C_i] \stackrel{\text{def}}{=} [\sigma_{\nu_*, l_i}(X_i)(C_i) \stackrel{\Pi, \mu, M}{\Rightarrow} \emptyset] \quad (2)$$

$$[l_i :: X_i => C_i] \stackrel{\text{def}}{=} [\sigma_{\nu_*, l_i}(X_i)(\# \Pi^* _ \Pi^* \#) \stackrel{\Pi, \mu, M}{\Rightarrow} \sigma_{\nu_*, l_i}(X_i)(C_i)] \quad (3)$$

$$[l_i :: X_i <= C_i] \stackrel{\text{def}}{=} [\sigma_{\nu_*, l_i}(\pi_1^{-1}(\pi_1(X_i)))(C_i) \stackrel{\Pi, \mu, M}{\Rightarrow} \sigma_{\nu_*, l_i}(X_i)(C_i)] \quad (4)$$

$$[l_i :: X_i <=> C_i] \stackrel{\text{def}}{=} [\sigma_{\nu_*, l_i}(\pi_j^{-1}(\pi_j(X_i)))(C_i) \cup \sigma_{\nu_*, l_i}(X_i)(\# \Pi^* _ \Pi^* \#) \stackrel{\Pi, \mu, M}{\Rightarrow} \sigma_{\nu_*, l_i}(X_i)(C_i)]. \quad (5)$$

In contrast to [4], the generalized postconditions specify now immediately the successful rule applications (like S_i later in [4]).

Since the original GTWOL [4], context restriction rules have had both licensing and restricting functions because of the longest application principle [4,5]. While the problematic left-arrow rules with empethesis [11] were addressed [4], the double function of context restriction rule $[(\mathbf{a} \cup \mathbf{a}:\mathbf{b})^* => \mathbf{c_d}]$ restricted the occurrences of such substrings as ϵ , \mathbf{a} and \mathbf{aa} . The currently updated GTWOL contains a *default core* GEN_{core} of two rules: rule $[1 :: \Pi => \emptyset]$ says that every symbol in strings needs to be licensed, and rule $[1 :: I^* => _]$ says that all substrings consisting of identity pairs are licensed. The latter default rule is now in an intended conflict with $[1 :: \mathbf{a}^* => \mathbf{c_d}]$.

Coherent Intersection An important aspect of GTWOL is how it combines rules. In the classical Two-Level Grammar, rules are compiled in separation and then combined under intersection, whereas GTWOL can combine rules *before* they are compiled. The operation \pitchfork under which the rules are combined is introduced in [5], and it is called *coherent intersection*.

$$[W_1 \stackrel{\Pi, \mu, M}{\Rightarrow} W'_1] \pitchfork [W_2 \stackrel{\Pi, \mu, M}{\Rightarrow} W'_2] \stackrel{\text{def}}{=} [(W_1 \cup W_2) \stackrel{\Pi, \mu, M}{\Rightarrow} ((W_1 \cap W'_1) \cup (W_2 \cap W'_2))] \quad (6)$$

Let G be a collection of GTWOL rules that use alphabet Π . When all rules are combined under the coherent intersection, the grammar reduces to a single generalized restriction $W \stackrel{\Pi, \mu, M}{\Rightarrow} W'$ that returns the language described by G . This language is denoted by GEN_G .

Coherent intersection implements conflict resolution for various kinds of arrow conflicts [12,4,5]. In addition, two further resolution strategies follow from the definition of $\nu_{*, l}$: conflicts between embedded rule applications are resolved on the basis of the *longest application* principle [4] and *disjunctive ordering* of the levels [5]. Disjunctive ordering uses alternative diamonds $\diamond_1, \dots, \diamond_s$. The

disjunctive level denoted by \circ_1 is the least general one. Rules at level strictly greater than 1 use several alternative diamonds. However, partially overlapping applications are not resolved automatically and rules with shorter applications cannot override rules with strictly longer applications. This is where GTWOL will continue to mature.

Most GTWOL rules are stored at the level 1. Therefore, we can abbreviate such rules by leaving out their level specifications.

Bimorphisms Defined by GTWOLs Bimorphisms [13] are a useful notion that can be combined with generalized restriction [5]. Let Σ_1, Σ_2 and Π be alphabets. A *bimorphism* is a triple (ψ_1, P, ψ_2) where $\psi_1 : \Pi^* \rightarrow \Sigma_1^*$ is the *input homomorphism*, $P \subseteq \Pi^*$ is the *pivot*, and $\psi_2 : \Pi^* \rightarrow \Sigma_2^*$ is the *output homomorphism*. The transformation relation $\beta(P) \subseteq \Sigma_1^* \times \Sigma_2^*$ computed by bimorphism is defined as $\beta(P) = \{(\psi_1(w), \psi_2(w)) \mid w \in P\}$.

Let $\text{GEN}_G \subseteq \Pi^*$ be a language described by a two-level grammar. According to bimorphism $(\pi_1, \text{GEN}_G, \pi_2)$, this language defines a regular relation $\beta_1(\text{GEN}_G)$ where $\beta_1(P) = \{(\pi_1(w), \pi_2(w)) \mid w \in P\}$ [9,4].

4 Reduction of Replace Rules into Two-Level Grammars

In the literature, a diverse variety of algorithms have been proposed as solutions to compilation of oriented, inverted, directed, parallel, and ranked replacement and marking rules [6,7,1,14]. In order to integrate different rule types and their compilation methods, we relate them to Generalized Two-Level Grammar that provides a good basis for representation of conditions of individual rules.

Centers The heart of a usual replacement rule is the *description of change*, or the *center*, that consists of *two regular languages*, $U \subseteq \Sigma^*$ and $Y \subseteq \Sigma^*$, meaning that a substring in U will be replaced disjunctively with replacements that are picked from set Y . The separate description of U and V is motivated by the usual rule formats in production systems and it may be easier to read. Some rules *e.g.* in Generative Phonology contain backreferences that are normally expressed with feature variables. According to Kaplan and Kay [11], such rules could be split into a number of subrules.

However, it is arguable [15] that if the centers are defined as *regular relations* we obtain a more expressive and useful definition that includes, for example, marking rules. Therefore, we will specify the center X_i directly as a same-length relation *i.e.* a language over Π . In fact, there are various ways to obtain an adequate X_i from languages U_i and Y_i , if needed. If X_i is obtained adequately, cross product $U_i \times Y_i$ equals to $\beta_1(X_i)$. Rules that contain a *list of centers* X_1, X_2, \dots, X_n reduce now to union $\cup_{i=1}^n X_i$. Moreover, the center of the *marking rules* [7,14] can be expressed easily as a subset of Π^* . For example, the description of change in the marking rule $[a^+ \rightarrow b \ e \ g \ \dots \ e \ n \ d]$ in XFST [14] corresponds to two-level center $0:b \ 0:e \ 0:g \ a^+ \ 0:e \ 0:n \ 0:d$.

Oriented Contexts Both replace (and marking) rules can be conditional [6,7,14] *i.e.* restricted to apply only in certain contexts. The context conditions of these rules can be reduced into GTWOL context conditions easily. For consistent presentation, assume that boundary markers $\cdot\#$ [14] and $\#\cdot$ are synonymous.

The previous implementations of replace rules express each context condition C_i as a language $\#L_i\#R_i\#$, where $L_i, R_i \subseteq \Sigma^*$. For convenience, each such context condition can be represented in a weaker form $C'_i \subseteq (\epsilon \cup \#)\Sigma^*_ \Sigma^*(\epsilon \cup \#)$ that is related to C_i by the following equivalence:

$$C_i = [(\epsilon \cup \#\Sigma^*)C'_i(\epsilon \cup \Sigma^*\#)] \cap (\#\Sigma^*_ \Sigma^*\#). \quad (7)$$

In contrast to two-level contexts that are subsets of $(\Pi \cup M)^*$, the replace rules restrict their context conditions to languages over $(\Sigma \cup M)^*$. This is due to the fact that these contexts have four possible orientations: *left-to-right*, *right-to-left*, *upward* and *downward*. If the context condition C_i of the replace rule is left-to-right (or right-to-left), it is interpreted as a combination of a look-a-head condition R_i (L_i) in the input string and a trailer condition L_i (R_i) in the output string. Conditions with either upward or downward orientation are simpler and they check either the input or the output string, respectively.

In Generative Grammar, the slash character / is conventionally used to separate the description of change and the context condition [16]. In the replace formalism [6], the specific form of this separator $s_i \in \{//, \backslash\backslash, ||, \backslash/\}$ indicates also the orientation of the context. The oriented context condition $s_i C_i$ corresponds to a two-level context condition

$$C_i = \begin{cases} \{x_1:x_2 \in \Pi^* \mid x \in d_0^{-1}(C_i) \wedge x_1:x \in \#\Pi^*_ \Sigma^*\# \wedge x:x_2 \in \#\Sigma^*_ \Pi^*\#\}, & \text{if } s_i = //'; \\ \{x_1:x_2 \in \Pi^* \mid x \in d_0^{-1}(C_i) \wedge x_1:x \in \#\Sigma^*_ \Pi^*\# \wedge x:x_2 \in \#\Pi^*_ \Sigma^*\#\}, & \text{if } s_i = \backslash\backslash'; \\ \{x:x_2 \in \Pi^* \mid x \in d_0^{-1}(C_i)\}, & \text{if } s_i = '||'; \\ \{x_1:x \in \Pi^* \mid x \in d_0^{-1}(C_i)\}, & \text{if } s_i = '\backslash/'. \end{cases} \quad (8)$$

The reduction lends itself for a simple finite-state implementation *e.g.* by using composition or a simpler *ad hoc* algorithm. Given the reductions (7) and (8), a typical weak replacement context condition such as $// \text{c_d}$ is considered a two-level context $\#\Pi^*\pi_2^{-1}(\text{c})_ \pi_1^{-1}(\text{d})\Pi^*\#$.

The subsets of $\#\Pi^*_ \Pi^*\#$ are obviously closed under the Boolean operations. As we have now reduced all context conditions into these sets, we can combine contexts with different orientations under intersection, asymmetric difference and symmetric difference. Accordingly, we capture more than the usual possibilities [14] with considerable ease.

Two-Level Operators for Replace Modes When the center X_i and the context condition C_i are both in the two-level format, it is natural to introduce a flexible rule syntax for parallel rules. On one hand, centers and context conditions can both be combined with Boolean operations. On the other, the extended generalized restrictions obtained from each parallel rule can be combined under

coherent intersection. Parallel rules can be indicated in the rule formalism in different ways. One possibility is the XFST notation:

$$X_1 \text{ op}_1 C_1,, X_2 \text{ op}_2 C_2,, \dots,, X_n \text{ op}_n C_n. \quad (9)$$

In XFST, the rule operators are used to indicate the mode of application. Alternative operators include $(->)$, $->$, $<-$, $<->$, $@->$, $->@$, $@>$, and $>@$. These indicate respectively the optional, obligatory, inverse, bidirectional, longest left-to-right, longest right-to-left, shortest left-to-right, and shortest right-to-left replacement modes.

In order to account for different replacement modes compactly, it is useful to understand what aspects they have in common and which mode could be used as a primary notion for obtaining the others.

4.1 Overlapping vs. Non-Overlapping Applications

In GTWOL, rules such as $[1 :: \mathbf{a:b=>c_d}]$ are actually very similar to optional replacement rules [4,5]. Provided that the rule neither overlaps nor interacts with itself or any other rule than the default rules, the semantics of context restriction actually coincides with optional replacement. However, a self-overlapping context restriction $[1 :: \mathbf{a:b a:b=>_}]$ and optional replace $\mathbf{aa(->)bb}$ are not interchangeable (consider *e.g.* the input \mathbf{aaa}). And due to the overlaps, context restriction rules $[1 :: \mathbf{a:b=>x_x}]$ and $[1 :: \mathbf{x a:b x=>_}]$ do not differ when considered in isolation [2]: both would accept the symbol-pair string $\mathbf{x a:b x a:b x}$. However, the contributions of these rules differ under coherent intersection because the latter rule has a longer center.

Double-arrow rules are the classical way to express obligatoriness in two-level grammars. They involve a right-arrow rule and a left-arrow rule. However, the resulting notion of obligatoriness is quite strict (denoted by **M1**), because the combination of such left-arrow rules as $[1 :: \mathbf{A:a B:p<=_}]$ and $[1 :: \mathbf{B:b C:c<=_}]$ rejects the input \mathbf{ABC} . The consequences $\mathbf{B:p}$ and $\mathbf{B:b}$ generate a conflict that is not solved automatically by the current GTWOL.

Kaplan and Kay [11] underlines that phonological rules do not normally rewrite their own output. This does not refer to overlapping simultaneous rule applications at the first place but such directed rewriting that does not advance monotonously in the original input string but resume, after an application, an earlier string position in the modified string. Anyway, overlapping applications does not belong to replace rules such as [1].

The interpretations of the optional replace rules [1] and right-arrow rules on one hand, and obligatory replace and double-arrow rules, on the other, will coincide if the center is free from self-overlaps and self-embeddings. Thus, replacement rules should somehow be reduced to overlap-free GTWOL grammars.

4.2 A Bracketed GTWOL

We use term *Bracketed Generalized Two-level Grammar* (BGTWOL) to refer to a loosely characterized family of such GTWOL grammars that assume a non-empty sub-alphabet $B \subseteq I$ and use it to indicate bracketing in the strings of

language GEN_G . The default core GEN_{core} is now $[1 :: II \Rightarrow \emptyset] \uplus [1 :: \Sigma^* \Rightarrow _]$, because now alphabet $B \not\subseteq \Sigma$ is reserved for a special use and the user does not have a normal access to it.

Let G be a BGTWOL grammar. The language GEN_G described by grammar G is used as the pivot in bimorphism $(\psi_1, \text{GEN}_G, \psi_2)$ where $\psi_1(w) = \pi_1(d_B(w))$ and $\psi_2(w) = \pi_2(d_B(w))$. In this bimorphism, the grammar describes the regular relation $\beta_2(\text{GEN}_G)$ where $\beta_2(P) = \{(\pi_1(d_B(w)), \pi_2(d_B(w))) | w \in P\}$.

Bracketed Grammar Rules In addition to the disjunctive ordering of rules, BGTWOL involves another ranking mechanism that is based on bracket labels. It is, however, not really used before Section 6. For all $i = 1, 2, \dots$, let $X_i \subseteq (II \setminus B)^*$ and $C_i \subseteq \#(II \setminus B)^* _ (II \setminus B)^* \#$ be regular languages, and let $X'_i = \langle_{b_i} X_i \rangle_{b_i}$, $C'_i = d_B^{-1}(C_i)$, $\Delta_0 = (II \setminus B)^*$ and $\Delta_1 = \Delta_0(B_L \Delta_0 B_R \Delta_0)^*$.

BGTWOL supports some new rule types that include *bracketed coercion* $[l_i :: \langle_{b_i} X_i \rangle_{b_i} (\Leftarrow) C_i]$, *bracketed inverse coercion* $[l_i :: \langle_{b_i} X_i \rangle_{b_i} (\Leftarrow) C_i]$, *bracketed context restriction* $[l_i :: \langle_{b_i} X_i \rangle_{b_i} (\Rightarrow) C_i]$, *bracketed double-arrow* $[l_i :: \langle_{b_i} X_i \rangle_{b_i} (\Leftarrow \Rightarrow) C_i]$, *bracketed inverse double-arrow* $[l_i :: \langle_{b_i} X_i \rangle_{b_i} (\Leftarrow \Rightarrow) C_i]$, and *bracketed bidirectional double-arrow* $[l_i :: \langle_{b_i} X_i \rangle_{b_i} (\Leftarrow \Leftrightarrow) C_i]$. These operations are defined as follows:

$$[l_i :: X'_i (\Leftarrow) C_i] \stackrel{\text{def}}{=} [\sigma_{\nu^*, l_i}(\pi_1^{-1}(\pi_1(d_B(X'_i))))(C'_i \cap \# \Delta_1 _ \Delta_1 \#) \xrightarrow{\Pi, \mu, M} \emptyset] \quad (10)$$

$$[l_i :: X'_i (\Leftarrow) C_i] \stackrel{\text{def}}{=} [\sigma_{\nu^*, l_i}(\pi_2^{-1}(\pi_2(d_B(X'_i))))(C'_i \cap \# \Delta_1 _ \Delta_1 \#) \xrightarrow{\Pi, \mu, M} \emptyset] \quad (11)$$

$$[l_i :: X'_i (\Rightarrow) C_i] \stackrel{\text{def}}{=} [l_i :: X'_i \Rightarrow C'_i] \quad (12)$$

$$[l_i :: X'_i (\Leftarrow \Rightarrow) C_i] \stackrel{\text{def}}{=} [l_i :: X'_i (\Leftarrow) C_i] \uplus [l_i :: X'_i (\Rightarrow) C_i] \quad (13)$$

$$[l_i :: X'_i (\Leftarrow \Rightarrow) C_i] \stackrel{\text{def}}{=} [l_i :: X'_i (\Leftarrow) C_i] \uplus [l_i :: X'_i (\Rightarrow) C_i] \quad (14)$$

$$[l_i :: X'_i (\Leftarrow \Leftrightarrow) C_i] \stackrel{\text{def}}{=} [l_i :: X'_i (\Leftarrow \Rightarrow) C_i] \uplus [l_i :: X'_i (\Leftarrow \Rightarrow) C_i]. \quad (15)$$

Bracketed coercion bears functional similarity to surface coercion. It says intuitively that the center X_i that is non-embedded (*i.e.* $\# \Delta_1 _ \Delta_1 \#$) must not be left unbracketed in the specified contexts.

Applications of bracketed surface coercion can overlap one another, but even the first application suffices to reject the pair-string and is normally not cancelled by other GTWOL rules. Meanwhile, applications of bracketed context restrictions cannot be embedded or overlapping because X_i does not contain brackets. Accordingly, we can give for the operator a simpler, purely licensing definition that looks like a tautology but still contributes against the default rule $[1 :: II \Rightarrow \emptyset]$ under coherent intersection.

$$[l_i :: X'_i (\Rightarrow) C_i] \stackrel{\text{def}}{=} [\sigma_{\nu^*, l_i}(X'_i)(C'_i) \xrightarrow{\Pi, \mu, M} \sigma_{\nu^*, l_i}(X'_i)(C'_i)]. \quad (16)$$

Optional Replace Rules Yli-Jyrä and Koskenniemi [2] observe that parallel conditional optional replace rules can be represented using context restriction in GTWOL. In the current terms, the representation uses bimorphism

$(\psi_1, \text{GEN}_G, \psi_2)$ where the pivot GEN_G is obtained by changing the replace rules into *bracketed context restrictions*:

$$[X_1(->) C_1,, X_2(->) C_2,, \dots,, X_n(->) C_n] \stackrel{\text{def}}{=} \beta_2(\text{GEN}_{\text{core}} \uplus \uplus_{i=1}^n [1 :: \langle_1 X_i \rangle_1 (=>) C_i]) \quad (17)$$

Note that the brackets indicate the regions where a rule has been correctly applied. This is a quite different approach than the multiplicity of brackets that are used in [1] to indicate partial satisfaction of conditions for rule application. For example, pivot language GEN_G obtained from optional BGTWOL replace (that corresponds to rule $[\text{ab} (->) \text{x} // \text{ab} _ \text{b}]$ in Karttunen's [6] formalism)

$$[1 :: \text{a:x b:0} (->) \#II^* \pi_2^{-1}(\text{a b}) _ \pi_1^{-1}(\text{a}) II^* \#] \quad (18)$$

contains exactly the following mappings for input string **abababa**:

$$\begin{array}{lll} (19a) & (19b) & (19c) \\ \text{abababa} & \text{ab}\langle_1 \text{ab} \rangle_1 \text{aba} & \text{abab}\langle_1 \text{ab} \rangle_1 \text{a} \\ \text{abababa}, & \text{ab}\langle_1 \text{x0} \rangle_1 \text{aba}, & \text{abab}\langle_1 \text{x0} \rangle_1 \text{a}. \end{array} \quad (19)$$

Obligatory Replace Rules For the sake of compatibility to the Xerox calculus [14], it is desirable to pursue the semantics of obligatory conditional parallel replace such as in [6]. This can be done using *bracketed double-arrow* rules.

$$[X_1-> C_1,, X_2-> C_2,, \dots,, X_n-> C_n] \stackrel{\text{def}}{=} \beta_2(\text{GEN}_{\text{core}} \uplus \uplus_{i=1}^n [2 :: \langle_1 X_i \rangle_1 (<=>) C_i]). \quad (20)$$

Because the substrings undergoing a change are indicated by brackets, it is easy to enforce that a substring must be changed whenever the conditions for the replacement are met. This requirement is contributed by the bracketed coercion. Its disjunctive ordering level is now 2, because the default rule $[1 :: \Sigma^* \Rightarrow _]$ would cancel the effect of bracketed coercion $[1 :: \langle_1 X_i \rangle_1 (<=>) C_i']$ at level 1.

In inverse replacement (denoted by $\langle - \rangle$), the roles of the input and output strings are switched. The bidirectional obligatory replacement requires bracketed bidirectional double-arrow (*i.e.* $\langle \Rightarrow \rangle$).

5 Violable Mode Constraints

Although BGTWOL provides a solution to obligatory replacement through the bracketed double-arrow, we will now compare the solution to the new method of Yli-Jyrä and Koskenniemi [2]. For this purpose, we introduce some additional machinery.

5.1 Strict Preference Relations

A binary relation $T \subseteq I^* \times I^*$ is a *strict preference relation (SPR)* if it is irreflexive (thus not complete), transitive and antisymmetric. The following relations and their inverses are strict preference relations:

$$T_{\text{most}} = \{(\pi_1(w), \pi_2(w)) \mid w \in (B_L:0 \Sigma^* B_R:0 \cup \Sigma \cup B:B)^*\} \quad (21)$$

$$T_{\text{most}+} = \{(\pi_1(w), \pi_2(w)) \mid w \in (B_L:0 \Sigma^+ B_R:0 \cup \Sigma \cup B:B)^*\} \quad (22)$$

$$T_{\text{norep}} = \{(w, w') \mid w, w' \in I^* \wedge d_B(w) = d_B(w') \wedge w \notin (I^* B_L B_R B_L B_R I^*) \ni w'\} \quad (23)$$

$$T_{\text{lest}} = \{(w, w') \mid w, w' \in I^* \wedge d_B(w) = d_B(w') \wedge w \notin (I^* B I^*) \ni w'\} \quad (24)$$

$$T_{\text{lr}} = \{(vby, vau) \mid v, y, u \in I^* \wedge a \in \Sigma \wedge b \in B_L \wedge d_B(y) = d_B(au)\} \quad (25)$$

$$T_{\text{rl}} = \{(ybv, uav) \mid v, y, u \in I^* \wedge a \in \Sigma \wedge b \in B_R \wedge d_B(y) = d_B(ua)\} \quad (26)$$

$$T_{\text{rlong}} = \{(vau, vby) \mid v, u, y \in I^* \wedge a \in \Sigma \wedge b \in B_R \wedge d_B(y) = d_B(au)\} \quad (27)$$

$$T_{\text{rllong}} = \{(uav, ybv) \mid v, u, y \in I^* \wedge a \in \Sigma \wedge b \in B_L \wedge d_B(y) = d_B(ua)\} \quad (28)$$

$$T_{\alpha, B'} = \{(w, w') \mid w, w' \in I^* \wedge (d_{B \setminus B'}(w), d_{B \setminus B'}(w')) \in T_\alpha\}. \quad (29)$$

Let T be an SPR. According to T , element $x_1 \in I^*$ is interpreted strictly more preferable than $x_2 \in I^*$, i.e. $x_1 \prec x_2$, if and only if $(x_1, x_2) \in T$. For example, $T_{\text{most}+}$ compares only compatible bracketings and prefers those that have more markup:

$$\begin{aligned} aa \langle_1 ab \rangle_1 \langle_2 ab \rangle_2 a \prec \{ & aaab \langle_1 ab \rangle_1 a, aa \langle_1 ab \rangle_1 aba \} \prec aaababa; \\ \{ \langle_1 aa \rangle_1 \langle_1 aa \rangle_1, a \langle_1 aa \rangle_1 a \} & \prec aaaa. \end{aligned}$$

Let us prove that $T_{\text{most}+}$ is an SPR. Relation $T_{\text{most}+}$ removes at least one pair of brackets, but it can also remove more, or even all brackets. Therefore, the expressed relation is transitive, since for all $v, x, y \in I^*$, $(v, y) \in T_{\text{most}+}$ if $(v, x) \in T_{\text{most}+}$ and $(x, y) \in T_{\text{most}+}$. It is irreflexive and antisymmetric, since for all $(v, w) \in T_{\text{most}+}$, $|v| > |w|$. Thus, the relation of $T_{\text{most}+}$ is a SPR.

The union of two strict preference relations is not generally a strict preference relation since the result is not necessarily antisymmetric. Still, some preference relations can be combined under union.

All the SPRs defined in (21–29) are regular and easily implementable with finite-state transducers or bimorphisms. Typically the corresponding transducer contains only a few states.

5.2 Applications of Strict Preference Relations

The Method of Yli-Jyrä and Koskenniemi Yli-Jyrä and Koskenniemi [2] were inspired by the “matching” approach [17] used in selecting candidates that have minimal compatible set of constraint violations in Finite-State Optimality Theory. A somewhat similar approach has been used in [18]. In order to implement the method for parallel obligatory replacement [2], the minimality

constraint is inverted to obtain strings with maximal bracketing. The five steps of the resulting method in [2] are the following:

1. Prepare C_i (and compute X_i);
2. Compute $C'_i = d_B^{-1}(C_i)$ and $X'_i = \langle_1 X_i \rangle_1$;
3. Compute $\text{GEN}_G = [1 :: (H \setminus \Sigma) \Rightarrow \emptyset] \text{ \textcircled{R}} \text{ \textcircled{R}}_{i=1}^n [1 :: X'_i \Rightarrow C'_i]$;
4. Compute $D = \pi_1(\text{GEN}_G)$ and $D' = \{w_2 \mid w_1 \in D \wedge (w_1, w_2) \in T_{\text{most}+}\}$;
5. Compute $\text{GEN}'_G = \{w_1 : w_2 \in \text{GEN}_G \mid w_1 \notin D'\}$ and return $\beta_2(\text{GEN}'_G)$. (30)

Generalized Lenient Composition Jäger [3] defines a left-associative binary operator (**glc**) and controversially calls it *generalized lenient composition operator* although it rather addresses a problem with lenient composition than generalizes it. We add two variants: *inverse* one (denoted by **r-glc**) and *bidirectional* one (denoted by **b-glc**). The operators assume two operands: a candidate set $S \subseteq H^*$ and a strict preference relation $T \subseteq I^* \times I^*$, and they are defined as follows:

$$S \text{ glc } T \stackrel{\text{def}}{=} \{w \in S \mid \neg \exists w' (w' \in S \wedge (\pi_2(w), \pi_2(w')) \in T)\}; \quad (31)$$

$$S \text{ r-glc } T \stackrel{\text{def}}{=} \{w \in S \mid \neg \exists w' (w' \in S \wedge (\pi_1(w), \pi_1(w')) \in T)\}; \quad (32)$$

$$S \text{ b-glc } T \stackrel{\text{def}}{=} (S \text{ glc } T) \cap (S \text{ r-glc } T). \quad (33)$$

Now as we have slightly elaborated our formal machinery, we can express the compilation method of [2] as a two-step algorithm:

1. Compute $\text{GEN}_G = \text{GEN}_{\text{core}} \text{ \textcircled{R}} \text{ \textcircled{R}}_{i=1}^n [1 :: \langle_1 X_i \rangle_1 (\Rightarrow) C_i]$;
2. Compute $\beta_2(\text{GEN}_G \text{ r-glc } T_{\text{most}+})$. (34)

5.3 The Alternative Modes of Obligatoriness

Together with Kaplan and Kay [11], Yli-Jyrä and Koskenniemi [2] maintain that all other replacement modes are *subsets* of the relation described by the corresponding optional replacement (denoted by **Opt**), which contrasts to the approach of [6].

It is not trivial to describe how obligatory replacement restricts **Opt**. Three different approaches have already been presented in this paper (Sections 4.1, 4.2 and 5.2). These do not produce the results in general, and it is therefore at least fair to say a word about their differences. The replace relations corresponding to these modes form an inclusion order $\mathbf{M1} \subseteq \mathbf{M2} \subseteq \mathbf{M3} \subseteq \mathbf{Opt}$. The modes themselves can be described as follows:

M1 – Overlapping Synchronized Coercion Kaplan and Kay (page 357 of [11]) mention but do not elaborate a possibility of overlapping applications of obligatory rules. However, Section 4.1 and [2] point out that a GTWOL rule

can have overlapping applications. Due to this, a double arrow rule such as $[1 :: A:aB:p \cup B:bC:c \Leftrightarrow _]$ is in a self-conflict, which results into an over-constrained input-output mapping that fails to relate any output to input string ABC.

M2 – Non-Overlapping Coercion The bracketed double arrow of BGTWOL (Section 4.2) differs from the double arrow of GTWOL by using a bracketing that serves to avoid overlapping applications in every candidate mapping. Its left-arrow part is, however, surprisingly constrained since, for example, rule $[2 :: a:xb:0 (<=>) \#II^*\pi_2^{-1}(ab) _ \pi_1^{-1}(a)II^*\#]$ does not allow such mapping as $(abab<_1ab>_1a): (abab<_1x0>_1a)$.

As far as I can judge, Karttunen *et al.* [6,1,14] seem to implement this notion of obligatoriness into XFST when compiling the right-oriented rule $ab \ (->) \ x // ab \ _b$.² In particular, the definition 27 (the Replace component relation) in [6] cannot skip a center in a proper context without replacing it. Candidate $(abab<_1ab>_1a): (abab<_1x0>_1a)$ is not included to the result, because there is a non-overlapping substring (underlined) that should have been replaced with $x0$. The method ignores the fact that this additional change cannot be done (it would result into incorrect symbol-pair string $^*(ab<_1ab>_1<_1ab>_1a): (ab<_1x0>_1<_1x0>_1a)$) without altering the lower left context that was assumed by one of the changes.

M3 – Maximal Set of Non-Overlapping Changes This third notion of obligatoriness is represented by Section 5.2 and [2]. The subtle difference between the new method [2] and [6] was not recognized in [2] although it is a crucial part of backward compatibility. The semantics of the new method is illustrated considering the mappings of optional replace rule (18). Mappings (19b and 19c) are maximal candidates under the preference relation $T_{\text{most}+}$.

5.4 The GLC Approach to M2

Besides the bracketed double arrow, the new method of [2] can be modified to capture mode **M2**. The solution is based on the idea of a *bracketed identity rule* where brackets B_2 are used to mark the valid replacement locations that are held back *i.e.* the applications of the rule $[1 :: <_2\pi_1(X_i)>_2 (<=>) C_i]$. The contribution of this is to make the candidate set more dense under $T_{\text{most}+}$. SPR T_{lest,B_2} prefers candidates that do not contain B_2 . Pivot GEN_G has always at least one candidate without brackets B_2 .

$$[X_1->C_1, \dots, X_i->C_n] \stackrel{\text{def}}{=} \beta_2(\text{GEN}_G \text{ r-glc } (T_{\text{most}+} \cup T_{\text{lest},B_2})) \quad (35)$$

where $\text{GEN}_G = \text{GEN}_{\text{core}} \uplus \uplus_{i=1}^n [1 :: <_1X_i>_1 \cup <_2\pi_1(X_i)>_2 (<=>) C_i]$.

² It may be helpful to remark that [19] and [11] compile *directed* rewriting rules, see the discussion in [6]. Therefore they do not belong here.

Example The set GEN_G contains the following candidates for the unbracketed input abababa :

$$\left\{ \begin{array}{llll} \text{abababa} & \text{ab}\langle_2\text{ab}\rangle_2\text{aba} & \text{abab}\langle_2\text{ab}\rangle_2\text{a} & \text{ab}\langle_2\text{ab}\rangle_2\langle_2\text{ab}\rangle_2\text{a} \\ \text{abababa}, & \text{ab}\langle_2\text{ab}\rangle_2\text{aba}, & \text{abab}\langle_2\text{ab}\rangle_2\text{a}, & \text{ab}\langle_2\text{ab}\rangle_2\langle_2\text{ab}\rangle_2\text{a}, \\ & \text{ab}\langle_1\text{ab}\rangle_1\text{aba} & \text{abab}\langle_1\text{ab}\rangle_1\text{a} & \text{ab}\langle_2\text{ab}\rangle_2\langle_1\text{ab}\rangle_1\text{a} \\ & \text{ab}\langle_1\text{x0}\rangle_1\text{aba}, & \text{abab}\langle_1\text{x0}\rangle_1\text{a}, & \text{ab}\langle_2\text{ab}\rangle_2\langle_1\text{x0}\rangle_1\text{a} \end{array} \right\}. \quad (36)$$

The set of all *maximal* bracketings in GEN_G is obtained using strict preference relation $T_{\text{most}+}$ that ignores the bracket labels when comparing bracketed strings:

$$\begin{array}{l} \text{GEN}_G \text{ r-glc } T_{\text{most}+} \\ \cap \\ d_B^{-1}(\pi_1^{-1}(\text{abababa})) \end{array} = \left\{ \begin{array}{l} \text{ab}\langle_2\text{ab}\rangle_2\langle_2\text{ab}\rangle_2\text{a} \quad \text{ab}\langle_1\text{ab}\rangle_1\text{aba} \quad \text{ab}\langle_2\text{ab}\rangle_2\langle_1\text{ab}\rangle_1\text{a} \\ \text{ab}\langle_2\text{ab}\rangle_2\langle_2\text{ab}\rangle_2\text{a}, \text{ab}\langle_1\text{x0}\rangle_1\text{aba}, \text{ab}\langle_2\text{ab}\rangle_2\langle_1\text{x0}\rangle_1\text{a} \end{array} \right\}. \quad (37)$$

The set of candidates without identity rule applications is obtained with an additional preference transducer T_{lest,B_2} :

$$\begin{array}{l} \text{GEN}_G \text{ r-glc } T_{\text{lest},B_2} \\ \cap \\ d_B^{-1}(\pi_1^{-1}(\text{abababa})) \end{array} = \left\{ \begin{array}{l} \text{ab}\langle_1\text{ab}\rangle_1\text{aba} \quad \text{abab}\langle_1\text{ab}\rangle_1\text{a} \\ \text{ab}\langle_1\text{x0}\rangle_1\text{aba}, \quad \text{abab}\langle_1\text{x0}\rangle_1\text{a} \end{array} \right\}. \quad (38)$$

There is only one candidate that remains in the intersection of these sets.

$$\begin{array}{l} \text{GEN}_G \text{ r-glc } (T_{\text{most}+} \cup T_{\text{lest},B_2}) \\ \cap \\ d_B^{-1}(\pi_1^{-1}(\text{abababa})) \end{array} = \left\{ \begin{array}{l} \text{ab}\langle_1\text{ab}\rangle_1\text{aba} \\ \text{ab}\langle_1\text{x0}\rangle_1\text{aba} \end{array} \right\}. \quad (39)$$

Insertion Replaces It does not make sense to apply the obligatoriness constraint to insertion rules or, more generally, to rules where $\epsilon \in \pi_1(X_i)$. Because there could always be more insertions, no candidate would qualify as maximal according to T_{most} . Using SPR $T_{\text{most}+}$ instead has avoided this problem.

Sometimes it is desirable to make insertions only once at any position matching the context conditions. *E.g.* we may not want to limit insertions by consuming the material that might be rewritten by other parallel rules. Kempe and Karttunen [1] address the problem by providing a special strategy for one-time insertion. A similar strategy can be captured easily with SPR T_{norep} . After this constraint has been applied, it is natural to apply SPR T_{most} in order to get candidates with maximal sets of non-repeated insertions and other replacements.

$$[. X_i .] \rightarrow C_i \stackrel{\text{def}}{=} \beta_2(\text{GEN}_G \text{ r-glc } T_{\text{norep}} \text{ r-glc } (T_{\text{most}} \cup T_{\text{lest},B_2})). \quad (40)$$

Inverse and Bidirectional Replacement Inverse and bidirectional replacement [1] are extremely easy to implement. All what is needed is to use an adequate generalized lenient composition operator *i.e.* glc or b-glc .

Directed Replace It is possible to implement various directed replace operators [7] (and similar methods of [11,19]) using suitable strict preference relations.

$$X_i @-> C_i \stackrel{\text{def}}{=} \beta_2(\text{GEN}_G \text{ r-glc } (T_{\text{lr}} \cup T_{\text{lr}long})) \quad (41)$$

$$X_i ->@ C_i \stackrel{\text{def}}{=} \beta_2(\text{GEN}_G \text{ r-glc } (T_{\text{rl}} \cup T_{\text{rl}long})) \quad (42)$$

$$X_i @> C_i \stackrel{\text{def}}{=} \beta_2(\text{GEN}_G \text{ r-glc } (T_{\text{lr}} \cup T_{\text{lr}long}^{-1})) \quad (43)$$

$$X_i >@ C_i \stackrel{\text{def}}{=} \beta_2(\text{GEN}_G \text{ r-glc } (T_{\text{rl}} \cup T_{\text{rl}long}^{-1})). \quad (44)$$

Accordingly, we observe that using generalized restriction with BGTWOL gives an elegant and uniform approach for computing a large family of different replace (and marking) rules.

6 The Bipartite Approach

A New Design Pattern The method of [2] has a bipartite design that contains two main components: GEN_G and CON .

$$\beta_2(\text{GEN}_G \circ \text{CON}) = \beta_2(\text{GEN}_G \circ_1 T_1 \circ_2 T_2 \cdots \circ_m T_m). \quad (45)$$

The components are responsible for different kinds of tasks. GEN_G is the *candidate generator*, and CON is the lenient constraint component. The latter consists of *lenient constraints* T_1, T_2, \dots, T_n and left-associative *generalized lenient composition operators* $\circ_1, \circ_2, \dots, \circ_m \in \{\text{glc}, \text{r-glc}, \text{b-glc}\}$. Jäger [3] observes that lenient composition [20] can be expressed with generalized lenient composition.

The bipartite approach is very useful because it lends itself to many applications such as compilation of ranked rewriting rules and directed replacement rules. By encapsulating the context conditions of the replacements into GEN_G , the conditions are always observed in the generated candidates regardless of any strategic preferences. It is the task of CON to choose among alternative candidates, but it does not have to know about the internal structure of the candidate generator. By using strict preference relations, we obtain a uniform representation for various rule modalities.

Ranked Rules In Optimality Theory [21], the constraints are ranked. Similar ranking is possible also among parallel replacement rules. Various kinds of ranked rules have numberless applications beyond phonology and morphology.

Skut *et al.* [8] present a compiler for ranked left-to-right fixed-length replacement rules with upward-oriented contexts. A similar system of rules can be implemented easily in the current approach. First, we construct the bracketed GTWOL grammar corresponding to optional rules in such a way that brackets $<_i$ and $>_i$ occur in rules of rank i . The highest rank is now 1, and lowest n . The resulting bracketed relation, GEN_G , is constrained as follows:

$$\text{GEN}_G \text{ r-glc } T_{\text{lr},B_1} \text{ r-glc } T_{\text{lr},B_1 \cup B_2} \text{ r-glc } T_{\text{lr},B_1 \cup B_2 \cup B_3} \cdots \text{ r-glc } T_{\text{lr},B}. \quad (46)$$

In order to give preference to longest applications, strict preference relation $T_{lr,B'} \cup T_{lr\text{long},B'}$ could be used instead of $T_{lr,B'}$.

In [8], each rule rewrites a fixed-length substring. Our solution is more general since (i) the contexts in rules can be oriented and combined under Boolean operators, (ii) centers are not restricted to fixed-length substrings, and (iii) each rank can be shared by several parallel rules.

7 Conclusion

In the paper, we reviewed and extended the previously published 2-page description of the Yli-Jyrä and Koskenniemi method [2] for compiling parallel replace rules into transducers. Its relationship to the method of Kempe and Karttunen [1] is elaborated and discussed critically.

The background sections of this paper included an updated description of *Generalized Two-Level Grammars* (GTWOL). The semantics of GTWOL was defined, for the first time, using an *extended notion of generalized restriction*. In comparison to [11,19,1], the solution reduces considerably the number of different brackets needed to compile parallel replacement rules.

The main result in this work is to elaborate the *bipartite design pattern* that was employed implicitly in [2].

- Candidates are generated with a GTWOL grammar.
- Three forms of Jäger’s composition operator [3] (GLC) were employed.
- Strict preference relations account for obligatoriness, directionality and length-based preferences.

The design makes it easy to capture a *variety of rule application modes* without bothering about conditions of individual rules. Parallel replace rules can have even heterogeneous modes and the rules can be ranked.

In addition, the paper presented *three important notions of obligatoriness* and defined new compilation methods for each of them. The notion corresponding to the method of Kempe and Karttunen [1] was covered by two alternative solutions.

Acknowledgements

I am grateful for Måns Hulden for a better example rule (18) that is simpler than my original example. In addition, he suggested in September 2007 that the mode **M2** could be captured without generalized lenient composition. This inspired me to add the bracketed double arrow operator. Further comments by Dale Gerdemann helped me improve the presentation.

References

1. Kempe, A., Karttunen, L.: Parallel replacement in finite state calculus. In: 16th COLING 1996, Proc. Conf. Volume 2., Copenhagen, Denmark (1996) 622–627

2. Yli-Jyrä, A., Koskenniemi, K.: A new method for compiling parallel replacement rules. In Holub, J., Žďárek, J., eds.: *Implementation and Application of Automata, 12th International Conference, CIAA 2007, Revised Selected Papers*. Volume 4783 of LNCS., Springer (2007) 320–321
3. Jäger, G.: Gradient constraints in Finite State OT: The unidirectional and the bidirectional case. In: *Proceedings of FSMNLP 2001, an ESSLI Workshop, Helsinki (2001)* (35–40)
4. Yli-Jyrä, A., Koskenniemi, K.: Compiling generalized two-level rules and grammars. In: *Proceedings of FinTAL 2006*. LNAI (2006)
5. Yli-Jyrä, A.: Applications of diamonded double negation. In: *Proceedings of FSMNLP 2007, Potsdam, Germany (2008)*
6. Karttunen, L.: The replace operator. In: *33th ACL 1995, Proceedings of the Conference, Cambridge, MA, USA (1995)* 16–23
7. Karttunen, L.: Directed replace operator. In Roche, E., Schabes, Y., eds.: *Finite-state language processing, Cambridge, Massachusetts, A Bradford Book*. The MIT Press (1996) 117–147
8. Skut, W., Ulrich, S., Hammervold, K.: A flexible rule compiler for speech synthesis. In: *Proceedings of Intelligent Information Systems 2004, Zakopane, Poland (2004)*
9. Koskenniemi, K.: 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 (1983)
10. Karttunen, L., Beesley, K.R.: Two-level rule compiler. An additional documentation file on the CD-ROM supplement of Beesley and Karttunen (2003) (2003)
11. Kaplan, R.M., Kay, M.: Regular models of phonological rule systems. *Computational Linguistics* **20**(3) (1994) 331–378
12. Karttunen, L.: Finite-state constraints. In: *Proceedings of the International Conference on Current Issues in Computational Linguistics, Universiti Sains Malaysia, Penang, Malaysia (1991)*
13. Arnold, A., Dauchet, M.: Morphismes et bimorphismes d’arbres. *Theoretical Computer Science* **20** (1982) 33–93
14. Beesley, K.R., Karttunen, L.: *Finite state morphology*. CSLI Studies in Computational Linguistics. CSLI Publications, Stanford, CA, USA (2003)
15. Gerdemann, D., van Noord, G.: Transducers from rewrite rules with backreferences. In: *9th EACL 1999, Proceedings of the Conference*. (1999) 126–133
16. Chomsky, N., Halle, M.: *The Sound Pattern of English*. Harper and Row, New York (1968)
17. Gerdemann, D., van Noord, G.: Approximation and exactness in Finite-State Optimality Theory. In Eisner, J., Karttunen, L., Thériault, A., eds.: *SIGPHON 2000, Finite State Phonology*. (2000)
18. Eisner, J.: Directional constraint evaluation in optimality theory. In: *20th COLING 2000, Proceedings of the Conference, Saarbrücken, Germany (2000)* 257–263
19. Mohri, M., Sproat, R.: An efficient compiler for weighted rewrite rules. In: *34th ACL 1996, Proceedings of the Conference, Santa Cruz, CA, USA (1996)* 231–238
20. Karttunen, L.: The proper treatment of optimality in computational phonology. In: *Finite State Methods in Natural Language Processing*. (1998) 1–12
21. Prince, A., Smolensky, P.: *Optimality Theory: Constraint interaction in generative grammar*. Technical Report RuCCS TR-2, Rutgers University Center for Cognitive Science, New Brunswick, NJ (1993)