5

Bidirectional Optimality for Regular Tree Languages

Stephan Kepser

5.1 Introduction

Optimality theory (OT henceforth) has been introduced by Prince and Smolensky (1993) originally as a model for generative phonology. In recent years, this approach has been applied successfully to a range of syntactic phenomena, and it is currently gaining popularity in semantics and pragmatics as well. It is based on the idea that a mapping from one level of linguistic representation to another should be described in terms of rules and filters. The novel contribution of OT is that filters – or, synonymously, constraints – are ranked and violable. Thus the result of a rule-based generation process may still be acceptable although it violates certain constraints as long as other results violate more constraints or constraints that are higher ranked.

In other words, the rules generate a set of candidates that are competitors. On this set, the constraints are applied in the order of their ranking starting with the highest ranked constraint. A candidate may violate a constraint more than once. The application of the highest ranked constraint assigns each candidate the number of violations of that constraint. Some of the candidates are now optimal with respect to this constraint in the sense that they violate the constraint the fewest times. These, and only these, are retained for the next round of constraint application. In each round, the current constraint is applied to the set of candidates remaining from the previous rounds. And only

Proceedings of Formal Grammar 2004.
Paola Monachesi, Gerhard Jäger,
Gerald Penn and Shuly Wintner (eds.).

those candidates that are optimal with respect to the current constraint make it into the next round. In the end, after applying all constraints, a set of candidates is reached which is optimal with respect to the given ranking of the constraints. The method is therefore comparable to a high jump competition in athletics.

Frank and Satta (1998) show that certain classes of OT-systems can be handled by finite state techniques. Their approach is influenced by ideas from computational phonology, the original field of application for OT. In this view, the generation of candidates is a relation on strings, and this relation is defined by a finite state transducer. In order to also render constraints by finite state automata, two restrictions have to be made. The first one is that constraints have to be binary, that is to say, each constraint assigns each candidate either 0 or 1. The second restriction demands constraints to be *output* constraints. An output constraint is a constraint that assigns a number to a candidate pair purely on the base of its output. Under these restrictions, constraints can be rendered as regular string languages over the output. The aim of the paper by Frank and Satta (1998) is to provide a modularity result for the complexity of an OT-system in the following sense. Suppose that the set of candidates is given by a finite state transducer and all constraints are expressable by regular languages. Then the whole OT-system can be rendered by finite state techniques and is no more complex than its components. The success of the approach by Frank and Satta is based on well-known closure properties of regular string languages.

The work by Frank and Satta was extended into two diverging directions. Based on the observation that natural language syntax and semantics have trees as their underlying data structures and not strings Wartena (2000) and Kepser and Mönnich (2003) propose ways to extend the approach by Frank and Satta to tree languages. Wartena (2000) shows that the original results for strings can be extended straight forwardly to trees, since the closure properties for regular string languages needed by Frank and Satta also hold for regular tree languages. Observing that there are certain non-regular phenomena in some natural languages (see, e.g., (Shieber, 1985)) Kepser and Mönnich (2003) extend the result by Wartena to linear context-free tree languages. In their work, the generator is split into a source for input trees defined by means of a linear context-free tree grammar and a relation between input trees and output trees defined by a linear tree transducer. Constraints are defined by monadic second-order formulae or, equivalently, regular tree languages. Their modularity result is based on closure properties of linear context-free tree languages also

shown in that paper.

The second direction concerns the notion of optimality. All of the above described approaches are *unidirectional* in the sense that they describe ways to find optimal output for a given input. This view is apparently generation driven. Blutner (2000) points out that in particular in semantics and pragmatics unidirectional optimality may not suffice. The optimal interpretation of an utterance is obtained by an interplay between the generation process on the speaker side and the parsing process on the hearer side. Blutner therefore introduces the notion of *bidirectional* optimality theory. Formal properties of bidirectional OT are studied by Jäger (2002, 2003). He shows that the modularity result of Frank and Satta extends to bidirectional OT-systems of regular string languages. Jäger (2003) also shows that for bidirectional OT-systems the restriction to binary constraints is essential to achieve the modularity result.

Jäger states that the construction for bidirectional optimality of string languages in the earlier paper extends to bidirectional optimality of regular tree languages, if some automaton or tree transducer representing the Cartesian product of two regular tree languages can be provided. Actually, the proofs Jäger presents are that general in nature that they need not change for the case of regular tree languages. The present paper closes this gap. Cartesian products of regular tree languages can be defined by means of so-called tree tuple automata (Comon et al., 1997). And tree tuple automata can be integrated into the finite state construction used to compute bidirectionally optimal pairs. Thus the modularity result by Jäger (2002) for bidirectional optimality extends to the case of regular *tree* languages.

For obvious reasons the present paper follows Jäger (2002) very tightly quoting it frequently.

5.2 Preliminaries

Regular Tree Grammars

A *ranked alphabet* (or *ranked operator domain*) Σ is an indexed family $\langle \Sigma_n \rangle_{n \in N}$ of disjoint sets. A symbol f in Σ_n is called an *operator of rank* n. If $n = 0$, then f is also called a constant. For a ranked alphabet Σ, the set of *trees over* Σ (or Σ-*trees* or *terms over* Σ), denoted T_Σ is the smallest set of strings over $\Sigma \cup \{(,)\}$ such that $\Sigma_0 \subseteq T_\Sigma$ and, for $n \geq 1$, if $f \in \Sigma_n$ and $t_1, \ldots, t_n \in T_\Sigma$, then $f(t_1, \ldots, t_n) \in T_\Sigma$. A subset of T_Σ is called a *tree language* over Σ.

Any set M can be interpreted as a signature in which all of the elements of M are constants. Thus $T_{\Sigma \cup M}$ denotes the set of trees over Σ

and M. Let $X = \{x_1, x_2, x_3, \dots\}$ be an infinite set (of variables). Then $T_\Sigma(x_1, \dots, x_n) = T_{\Sigma \cup \{x_1, \dots, x_n\}}$ denotes the set of trees over Σ and additional variables $\{x_1, \dots, x_n\}$. From the point of view of a signature, a "variable" is a constant. If $t \in T_\Sigma(X)$ then we also write $t(x_1, \dots, x_n)$ to indicate that the variables of t are a subset of the set $\{x_1, \dots, x_n\}$. Let Σ and Ω be two signatures, let $t(x_1, \dots, x_n) \in T_\Sigma(x_1, \dots, x_n)$, and let $t_1, \dots, t_n \in T_\Omega$. Then $t(t_1, \dots, t_n) \in T_{\Sigma \cup \Omega}$ is the result of simultaneously replacing each occurrence of x_j in $t(x_1, \dots, x_n)$ by t_j (with $1 \leq j \leq n$).

Now we define the notion of a regular tree grammar.

Definition 1 A *regular tree grammar* is a quadruple $G = (\Sigma, \mathcal{F}, S, P)$ where

Σ	is a finite ranked alphabet of *terminals*,
$\mathcal{F}$	is a finite set of *nonterminals* or *function symbols*, disjoint with Σ,
$S \in \mathcal{F}$	is the *start symbol*, and
P	is a finite set of productions (or rules) of the form $F \to t$, where $F \in \mathcal{F}$ and $t \in T_{\Sigma \cup \mathcal{F}}$.

For a regular tree grammar $G = (\Sigma, \mathcal{F}, S, P)$ the derivation relation is given as follows. Let $s_1, s_2 \in T_{\Sigma \cup \mathcal{F}}$. We say $s_1 \Rightarrow s_2$ if and only if there is a production $F \to t$ and F is a leaf node of s_1. Tree s_2 is obtained from s_1 by replacing F with the tree t. As usual, $\stackrel{*}{\Rightarrow}$ stands for the reflexive-transitive closure of $\Rightarrow$. For a regular tree grammar G, we define $L(G) = \{t \in T_\Sigma \mid S \stackrel{*}{\Rightarrow} t\}$. $L(G)$ is called the *tree language* generated by G.

Tree Automata and Tree Transducers

For regular tree languages there exists an automaton model that corresponds to finite state automata for regular string languages. Let Σ be a signature. A *frontier-to-root tree automaton* is a triple $\mathcal{A} = (Q, F, \delta)$ where Q is a finite set of *states*, $F \subseteq Q$ a set of *final* states and δ is a finite transition relation. Each transition has the form

$$f(q_1, \dots q_n) \to q$$

where $f \in \Sigma_n$, $q, q_1, \dots, q_n \in Q$. On an intuitive level, a frontier-to-root tree automaton labels the nodes in a tree with states starting from the leaves and going to the root. Suppose n is a node in the tree and f is the k-ary function symbol at node n and the k daughters of n are already labelled with states $q_1, \dots, q_k$, and furthermore $f(q_1, \dots, q_k) \to q$ is a transition of $\mathcal{A}$, then node n can be labelled with state q. A tree is accepted if the root can be labelled with a final state.

We will now report some results about the theory of regular tree lan-

guages. For more information, consult the work by Gécseg and Steinby (1984, 1997). A tree language L is regular if and only if there is a tree automaton that accepts L. Regular tree languages are closed under union, intersection, and complement. There are corresponding constructions for tree automata. And finally, for every tree automaton accepting language L there exists a tree automaton also accepting L which has a single final state.

Tree automata can be generalised to automata that transform one tree into another, so-called tree transducers. The following exposition on tree transduction is taken from (Gécseg and Steinby, 1997). Let Σ and Ω be two signatures. A binary relation $\tau \subseteq T_\Sigma \times T_\Omega$ is called a *tree transformation*. A pair $(s,t) \in \tau$ is interpreted to mean that τ may transform s into t. We can speak of compositions, inverses, domains, and ranges of tree transformations as those of binary relations. We will now define frontier-to-root tree transducers.

Definition 2 A *frontier-to-root tree transducer* (or F-transducer) consists of a quintuple $\mathcal{A} = (\Sigma, \Omega, Q, P, F)$ where Σ and Ω are signatures; Q is a finite set of *states*, each element of Q is a unary function; $F \subseteq Q$ is the set of *final states*; and P is a finite set of *productions* of the following type:

$$f(q_1(x_1), \ldots, q_m(x_m)) \to q(t(x_1, \ldots, x_m))$$

where $f \in \Sigma_m$, $q_1, \ldots, q_m, q \in Q$, $t(x_1, \ldots, x_m) \in T_\Omega(x_1, \ldots, x_m)$.

The transformation induced by an F-transducer is defined as follow. We write QT_Ω for the set $\{q(t) \mid q \in Q, t \in T_\Omega\}$ and regard QT_Ω as a set of constants. Let $s, t \in T_{\Sigma \cup QT_\Omega}$ be two trees. It is said that t can be obtained by a direct derivation from s in $\mathcal{A}$ iff t can be obtained from s by replacing an occurrence of a subtree $f(q_1(t_1), \ldots, q_m(t_m))$ (with $f \in \Sigma_m, q_1, \ldots, q_m \in Q, t_1, \ldots, t_m \in T_\Omega$) in s by $q(t(t_1, \ldots, t_m))$, where $f(q_1(x_1), \ldots, q_m(x_m)) \to q(t(x_1, \ldots, x_m))$ is a production from P. If s directly derives t in $\mathcal{A}$ then we write $s \Rightarrow_\mathcal{A} t$. The reflexive transitive closure $s \Rightarrow^*_\mathcal{A} t$ is the derivation relation.

Intuitively, an F-transducer traverses a tree s from the leaves to the root rewriting it at the same time. In a single derivation step we consider a node n in s with label f where all the daughter nodes are already transformed into trees of T_Ω and each daughter node is in some state q_i. Then we replace the subtree of node n with the tree t from the production where the place holder variables of t are replaced by the trees of the daughter nodes of n. The root of this subtree is put into state q.

The relation

$$\tau_{\mathcal{A}} = \{(s,t) \mid s \in T_\Sigma, t \in T_\Omega, s \Rightarrow^*_{\mathcal{A}} q(t) \text{ for some } q \in F\}$$

is the transformation relation induced by $\mathcal{A}$. A relation $\tau \subseteq T_\Sigma \times T_\Omega$ is an F-*transformation* if there exists an F-transducer $\mathcal{A}$ such that $\tau = \tau_{\mathcal{A}}$. For a tree language $L \subseteq T_\Sigma$ we define $\mathcal{A}(L) = \{t \in T_\Omega \mid \exists s \in L \text{ with } (s,t) \in \tau_{\mathcal{A}}\}$.

A production $f(q_1(x_1), \ldots, q_m(x_m)) \to q(t(x_1, \ldots, x_m))$ is called *linear* if each variable $x_1, \ldots, x_m$ occurs at most once in t. An F-transducer is linear if each production is linear. We denote a linear F-transducer by LF-transducer. We will make use of the following results about LF-transducers.

Proposition 1 *LF-transducers are closed under composition. The classes of regular tree language is closed under LF-transductions. The domain and range of an LF-transducer is a regular tree language. For every regular tree language L there is an LF-transducer ι such that $Dom(\iota) = Rng(\iota) = L$ and ι is the identity on L.*

These results can be found, e.g., in (Gécseg and Steinby, 1984, 1997). If $\mathcal{A}_1$ and $\mathcal{A}_2$ are two LF-transducers, we write $\mathcal{A}_1 \circ \mathcal{A}_2$ for the composition of first $\mathcal{A}_1$ and then $\mathcal{A}_2$. Let L_1 and L_2 be two tree languages. A binary relation $R \subseteq L_1 \times L_2$ is called *rational* if there exists an LF-transducer $\mathcal{A}$ such that $\tau_{\mathcal{A}} = R$.

LF-transducers define relations between tree languages where there is a strong connection between the input trees and the output trees. Obviously, an output tree is constructed on the base of a structural decomposition of the input tree. As stated in the introduction, we also need an automaton or a transducer representation of the Cartesian product of arbitrary regular tree languages. In a Cartesian product, every element of the input tree language is related to every element of the output tree language. Hence there is in general no structural relation between an input tree and an output tree. Therefore LF-transducers are not capable of defining the Cartesian product of two regular tree languages, as already observed by Jäger (2002).

But the Cartesian product of two (or more) regular tree languages can be defined by means of *tree tuple automata*. Comon et al. (1997, Sect. 3.2) describe three classes of tree tuple automata two of which can be used for the definition of a Cartesian product. The simplest class, which suffices already for our purposes, is based on pairs of automata. Let L_1 and L_2 be two regular tree languages, and let $\mathcal{A}_1$ be a bottom-up tree automaton accepting L_1, and $\mathcal{A}_2$ a bottom-up tree automaton accepting L_2. Define for any two trees s, t that $(s,t) \in (\mathcal{A}_1, \mathcal{A}_2)$ iff

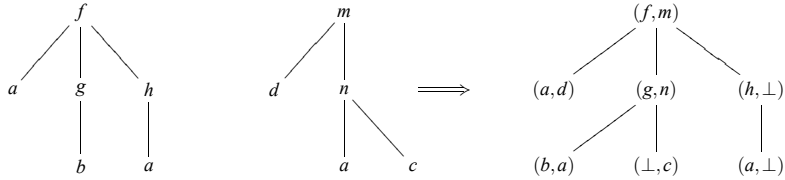

FIGURE 1 Coding two trees in a single one.

$s \in \mathcal{A}_1$ and $t \in \mathcal{A}_2$. Then, obviously, $(\mathcal{A}_1, \mathcal{A}_2)$ defines the relation $L_1 \times L_2$. This relation is called $Rec_\times$ in (Comon et al., 1997).

A more powerful definition of tree tuple automata is given, if the relation is expressed by a single automaton instead of a pair of automata. To make this approach work, two trees have to be coded in a single one. Let Σ and Ω be two signatures. We define trees with labels as pairs of $(\Sigma \cup \{\bot\} \times \Omega \cup \{\bot\})$ where $\bot$ is a new (padding) symbol not contained in Σ or Ω. For a pair of trees $(s,t) \in (T_\Sigma \times T_\Omega)$ we define inductively their coding $[s,t]$ by

$$[f(t_1,\ldots,t_n), g(u_1,\ldots,u_m)] = \begin{cases} (f,g)([t_1,u_1],\ldots,[t_m,u_m],[t_{m+1},\bot],\ldots,[t_n,\bot]) & \text{if } n \geq m \\ (f,g)([t_1,u_1],\ldots,[t_n,u_n],[\bot,u_{n+1}],\ldots,[\bot,u_m]) & \text{if } m \geq n \end{cases}$$

Basically, the union of the tree domains is constructed and the nodes are labelled with pairs of labels. If one tree lacks a branch, then the padding symbol $\bot$ is used. See Figure 1 for an example.

Now, *Rec* is the set of relations $R \subseteq T_\Sigma \times T_\Omega$ such that $\{[s,t] \mid (s,t) \in R\}$ is accepted by a bottom-up tree automaton on the signature $(\Sigma \cup \{\bot\} \times \Omega \cup \{\bot\})$.

We quote some results on tree tuple automata (see Comon et al., 1997). $Rec_\times$ is strictly included in *Rec*. $Rec_\times$ and *Rec* are closed under union, intersection and complement.

There exists an extended definition of tree transducers by Comon et al. (1997), who introduce ϵ-rules. Extended tree transducers are strictly more powerful than standard tree transducers as defined above in the sense that every relation definable by a standard tree transducer is definable by an extended tree transducer. But there are relations definable by an extended tree transducer that cannot be defined by a standard tree transducer. These extended tree transducers, however, are not capable of defining arbitrary Cartesian products of regular tree languages. Therefore they cannot be used to substitute the above defined tree tuple automata.

Monadic Second-Order Logic

Monadic second-order logic (MSO) is an extension of first-order logic by set variables and quantification over set variables. MSO is quite a powerful logic. Many graph theoretical properties can be expressed in MSO, for example that a graph is a proper tree. Courcelle (1990) showed that if a relation is definable in MSO, then so is its transitive closure. For tree languages, it is known that a tree language is regular iff it is definable by an MSO formula (see, e.g., Gécseg and Steinby, 1984). Thus an MSO formula can be translated into a tree automaton accepting the same tree language.

5.3 Optimality Theory

We now turn to optimality theory. Let us start by making the notions of optimality theory more precise. In the general case, an OT-system consists of a binary relation **GEN** and a finite set of constraints that are linearly ordered. Constraints may be violated several times. So a constraint should be construed as a function from **GEN** into the natural numbers. Thus an OT-system assigns each candidate pair from **GEN** a sequence of natural numbers. The ordering of the elements of **GEN** that is induced by the OT-system is the lexicographic ordering of these sequences.

Definition 3 An *OT-system* is a pair $(\mathbf{GEN}, C)$ where **GEN** is a binary relation and $C = \langle c_1, \ldots, c_p \rangle, p \in \mathbb{N}$, is a linearly ordered sequence of functions from **GEN** to $\mathbb{N}$. Let $a, b \in \mathbf{GEN}$. We say a is more economical than b ($a < b$), if there is a $k \leq p$ such that $c_k(a) < c_k(b)$ and for all $j < k : c_j(a) = c_j(b)$.

Intuitively, an output o is optimal for some input i iff **GEN** relates o to i and o is optimal amongst the possible outputs for i. This is the notion of unidirectional optimality, which is obviously generation-driven. Bidirectional optimality reflects the fact that in semantics and pragmatics the relation between input (a form) and output (a meaning) can and perhaps should be regarded as an interplay between parsing optimality and generation optimality. Hence Jäger (2002), formalising ideas by Blutner (1998, 2000), defines *bidirectional* optimality as follows.

Definition 4 A form-meaning pair (f, m) is *bidirectionally optimal* iff

1. $(f, m) \in \mathbf{GEN}$,
2. there is no bidirectionally optimal (f', m) such that $(f', m) < (f, m)$,

3. there is no bidirectionally optimal (f, m') such that $(f, m') < (f, m)$.

Thus, checking whether a form-meaning pair is bidirectionally optimal requires simultaneous evaluation of form alternatives and meaning alternatives of this pair. This definition is not circular in cases where the ordering of pairs is well-founded. As was shown by Jäger (2002), the ordering of pairs given by the definition of an OT-system is indeed well-founded. Hence bidirectional optimality of OT-systems is not a circular notion.

The type of constraints considered in the literature on finite state OT and also used here is not quite as general as Definition 3 insinuates. Firstly, constraints have to be binary, i.e., a constraint assigns a candidate pair either the number 0 or 1. This restriction is not quite as severe as it may seem. Every constraint with an upper bound on the number of violations can be translated into a sequence of binary constraints.

Secondly, so-called markedness constraints are considered only. A markedness constraint is a constraint that either evaluates solely the input or solely the output.

Definition 5 Let $(\mathbf{GEN}, (c_1, \ldots, c_p))$ be an OT-system.
A constraint c_j is an *output markedness constraint* iff $c_j(i, o) = c_j(i', o)$ for all $(i, o), (i', o) \in \mathbf{GEN}$.
A constraint c_j is an *input markedness constraint* iff $c_j(i, o) = c_j(i, o')$ for all $(i, o), (i, o') \in \mathbf{GEN}$.

To gain a better understanding of how bidirectional optimality is evaluated on markedness constraints in a (finite) OT-system consider this example by Jäger (2002). The following text is a direct quote from (Jäger, 2002, p. 441f).
"Suppose $\mathbf{GEN} = \{1, 2, 3\} \times \{1, 2, 3\}$, and we have two constraints which both say 'Be small!' One of its instances applies to the input and one to the output. Thus formally we have

- $\mathcal{O} = (\mathbf{GEN}, C)$.
- $\mathbf{GEN} = \{1, 2, 3\} \times \{1, 2, 3\}$.
- $C = \langle c_1, c_2 \rangle$.
- $c_1(\langle i, o \rangle) = i$.
- $c_2(\langle i, o \rangle) = o$.

It follows from the way constraints are evaluated that $\langle i_1, o_1 \rangle <_{\mathcal{O}} \langle i_2, o_2 \rangle$ iff $i_1 \leq i_2, o_1 \leq o_2$, and $\langle i_1, o_1 \rangle \neq \langle i_2, o_2 \rangle$. Now obviously $\langle 1, 1 \rangle$ is bidirectionally optimal since both its input and its output obey the constraints in an optimal way. Accordingly, $\langle 1, 2 \rangle, \langle 2, 1 \rangle, \langle 1, 3 \rangle$, and $\langle 3, 1 \rangle$

are blocked, since they all share a component with a bidirectionally optimal candidate. There are still candidates left which are neither marked as optimal nor as blocked, so we have to repeat this procedure. Amongst the remaining candidates, $\langle 2,2\rangle$ is certainly bidirectionally optimal because all of its competitors in either dimension are known to be blocked. This candidate in turn blocks $\langle 2,3\rangle, \langle 3,2\rangle$. The only remaining candidate, $\langle 3,3\rangle$, is again bidirectionally optimal since all its competitors are blocked.[1] This example illustrates the general strategy for the finite case: Find the cheapest input-output pairs in the whole of **GEN** and mark them as bidirectionally optimal. Next mark all candidates that share either the input or the output component (but not both) with one of these bidirectionally optimal candidates as blocked. If there are any candidates left that are neither marked as bidirectionally optimal nor as blocked, repeat the procedure until **GEN** is exhausted." And this ended the quote.

The construction of an OT-system for tree languages looks in principle as follows. The generation relation **GEN** is expressed by an LF-transducer, i.e., it is a rational relation. The constraints are expressed by MSO-sentences either on the input or on the output. As stated before, an MSO-sentence can be translated into a regular tree language and an LF-transducer. It is important to note that a constraint in OT is violable, and this is also true for binary constraints. If some candidates fulfil the constraint, then all others are filtered out. But if all candidates miss the constraint, all of them pass through, because no candidate is relatively better than the others. Frank and Satta (1998) provide a construction called conditional intersection that gives a transducer implementing this violability. Wartena (2000) shows how to extend this construction to trees for unidirectional optimality. And Jäger (2002) provides the extension for optimality theory on regular string languages.

The proposal by Jäger (2002) can be transfered to regular tree languages in a rather direct manner. Let R be a rational relation and $L \subseteq Rng(R)$ be a regular tree language. The conditional intersection $R \uparrow L$ is defined as

$$R \uparrow L := (R \circ \iota_L) \cup (\iota_{Dom(R)-Dom(R\circ\iota_L)} \circ R).$$

For an input markedness constraint represented by $L \subseteq Dom(R)$ we set dually

$$R \downarrow L := (\iota_L \circ R) \cup (R \circ \iota_{Rng(R)-Rng(\iota_L\circ R)}).$$

[1] Bidirectional optimality thus predicts iconicity: the pairing of cheap inputs with cheap outputs is optimal, but also the pairing of expensive inputs with expensive outputs. See Blutner's (1998, 2000) papers for further discussion of this point.

These intersections relate individually optimal input and output pairs. But in bidirectional optimality we are looking for globally optimal pairs. Hence bidirectional intersection is defined as follows. Let R be a rational relation and c a binary markedness constraint. Let $*$ be an arbitrary constant, i.e., a tree consisting of a single node, the root, labelled with $*$. Then

$$R \Uparrow c := \begin{cases} R \circ \iota_{Rng((\{*\} \times Rng(R)) \uparrow c)} & \text{if } c \text{ is an output markedness constraint} \\ \iota_{Dom((Dom(R) \times \{*\}) \downarrow c)} \circ R & \text{else} \end{cases}$$

The construction works as follows. $\{*\} \times Rng(R)$ relates the regular tree language $\{*\}$ to any possible output of R, which is also a regular tree language. Conditional intersection with c gives $((\{*\} \times Rng(R)) \circ \iota_c) \cup (\iota_{\{*\} - Dom((\{*\} \times Rng(R)) \circ \iota_c)} \circ (\{*\} \times Rng(R)))$. By definition, this is equal to $(\{*\} \times (Rng(R) \cap c)) \cup ((\{*\} - Dom(\{*\} \times (Rng(R) \cap c))) \times Rng(R))$. Since tree tuple automata are closed under union and their domains and ranges are obviously regular tree languages, the construction is well defined. It is defined in such a way that either the left hand side or the right hand side of the $\cup$ is the empty relation, depending on whether $Rng(R) \cap c$ is empty or not.

If $Rng(R) \cap c$ is non-empty, then the above reduces to $\{*\} \times (Rng(R) \cap c)$. If $Rng(R) \cap c$ is empty, then this reduces to $\{*\} \times Rng(R)$. In either way, $Rng((\{*\} \times Rng(R)) \uparrow c)$ is the set of outputs of R that are optimal with respect to c, and it is a regular tree language. Thus $R \Uparrow c$ is a rational relation, and it is the set of pairs $(i, o) \in R$ that are optimal with respect to c. If c is an input markedness constraint, the dual statements are true.

Lemma 2 *Let $\mathcal{O} = (\mathbf{GEN}, C)$ be an OT-system, where $C = (c_1, \ldots, c_p)$ is a sequence of binary markedness constraints. Then*

$$(i, o) \in \mathbf{GEN} \Uparrow c_1 \ldots \Uparrow c_p$$

iff $(i, o) \in \mathbf{GEN}$ and there is no $(i', o') \in \mathbf{GEN}$ such that $(i', o') < (i, o)$.

The proof for this lemma is identical to the proof of Lemma 3 on page 443 of (Jäger, 2002).

The application of a sequence of constraints $C = (c_1, \ldots, c_p)$ to a rational relation R can be seen as an operator $C(R) := R \Uparrow c_1 \ldots \Uparrow c_p$ filtering out the globally optimal pairs. As a result, R is partitioned into three sets: $C(R)$, the set B of pairs that share one component with a pair of $C(R)$ (but not both), and the set U of pairs that share

no component with a pair of $C(R)$. The pairs that share one component with a pair in $C(R)$ are blocked, they cannot be bidirectionally optimal. But no statement can be made about the pairs in U. Some of them may be bidirectionally optimal, some may not. Thus, similarly to the toy example given before, the filtering procedure has to be applied to U. And this step of filtering and finding the unmarked pairs has to be iterated till R is exhausted.

Definition 6 Let $\mathcal{O} = (\mathbf{GEN}, C)$ be an OT-system. Define

$$\begin{aligned} X_0 &:= \emptyset, \\ X_{n+1} &:= X_n \cup C(\ \iota_{Dom(\mathbf{GEN})-Dom(X_n)} \circ \mathbf{GEN} \\ &\qquad \circ \iota_{Rng(\mathbf{GEN})-Rng(X_n)}\ ), \\ X &:= \bigcup_{n \geq 0} X_n. \end{aligned}$$

For every natural number n, X_{n+1} adds those pairs to X_n that are not blocked by X_n and optimal. Hence, X is the set of all bidirectionally optimal pairs.

Lemma 3 *Let $\mathcal{O} = (\mathbf{GEN}, C)$ be an OT-system. Then $(i, o) \in X$ iff $(i, o) \in \mathbf{GEN}$ and (i, o) is bidirectionally optimal.*

The proof of this lemma is given as the proof of Lemma 4, page 444 of (Jäger, 2002).

As Jäger (2002) remarks, the operation X_n is a cumulative definition of bidirectional optimality. And on the assumption that **GEN** and X_n are rational relations and the constraints are binary markedness constraints expressable as regular tree languages, the relation X_{n+1} is again a rational relation, i.e., expressable by finite state means. Since $\emptyset = \emptyset \times \emptyset$ is a rational relation on trees, bidirectional optimality of regular tree languages can be computed with finite state techniques provided the iteration of exhausting **GEN** terminates after a finite number of steps, i.e., there is a $k \in \mathbb{N}$ such that $X = X_k$.

Lemma 4 *Let $\mathcal{O} = (\mathbf{GEN}, C)$ be an OT-system with $C = (c_1, \ldots, c_p)$ where all c_i are binary markedness constraints. Then $X = X_{2^{p-1}}$.*

The proof of this lemma is given as the proof of Lemma 5, page 447 of (Jäger, 2002).

The following main theorem integrates the observations made in this section.

Theorem 5 *Let $\mathcal{O} = (\mathbf{GEN}, C)$ be an OT-system with $C = (c_1, \ldots, c_p)$ where all c_i are binary markedness constraints. Furthermore let* **GEN** *be a rational relation and all c_i be MSO-sentences. Then the set of bidirectionally optimal elements of* **GEN** *is again a rational relation.*

5.4 Conclusion

This paper extends the approach by Jäger (2002) for bidirectional optimality from regular string languages to regular tree languages. We have shown that if the generator of an OT-system consists of a linear frontier-to-root transducer and the constraints are expressed by MSO-sentences on either the input or the output, then the OT-system can be rendered by finite state tree automata and tree transducers. This implies that the complexity of the whole OT-system is not larger than the complexity of its most complex components.

The most important difference of the construction for tree languages as compared to the one for string languages can be found in the fact that we need two types of finte state devices for tree languages. Finite state string transducer are capable of expressing the Cartesian product of two regular string languages. Therefore the construction for the computation of optimal pairs needs finite state string transducers only. In the case of tree language, the situation is different. Cartesian products of arbitrary regular tree languages cannot be defined by means of LF-transducers, not even if one allows for ϵ-rules. Therefore we introduced tree tuple automata into the construction. On the other hand there exist relations of regular tree languages definable by LF-transducers – rational relations, as we called them – that cannot be defined by means of tree tuple automata. These are in particular relations where there is an intricate relationship between the input and the output trees. Thus tree tuple automata cannot replace the LF-transducers, and we need indeed both types of finite state tree devices.

A interesting and important question is whether this approach can be extended to more complex generation relations that are based context-free instead of regular tree grammars. These extensions seem rather difficult. It looks like there is a way to compute the globally optimal pairs using LF-transducers. But whether the recursion step involved in bidirectional optimality (the definition of the X_n) can be rendered by finite state means is doubtful.

References

Blutner, Reinhard. 1998. Lexical pragmatics. *Journal of Sematics* 15:115–162.

Blutner, Reinhard. 2000. Some aspects of optimality in natural language interpretation. *Journal of Semantics* 17:189–216.

Comon, Hubert, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. 1997. Tree automata techniques and applications. Available at:

`http://www.grappa.univ-lille3.fr/tata`. Release October, 1st 2002.

Courcelle, Bruno. 1990. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, vol. B, chap. 5, pages 193–242. Elsevier.

Frank, Robert and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics* 24:307–315.

Gécseg, Ferenc and Magnus Steinby. 1984. *Tree Automata*. Budapest: Akademiai Kiado.

Gécseg, Ferenc and Magnus Steinby. 1997. Tree languages. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages, Vol 3: Beyond Words*, pages 1–68. Springer-Verlag.

Jäger, Gerhard. 2002. Some notes on the formal properties of bidirectional optimality theory. *Journal of Logic, Language, and Information* 11:427–451.

Jäger, Gerhard. 2003. Recursion by optimization: On the complexity of bidirectional optimality theory. *Natural Language Engineering* 9(1):21–38.

Kepser, Stephan and Uwe Mönnich. 2003. A note on the complexity of optimality theory. In G. Scollo and A. Nijholt, eds., *Proceedings Algebraic Methods in Language Processing (AMiLP-3)*, pages 153–166.

Prince, Alan and Paul Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. Tech. Rep. RuCCTS-TR 2, Rutgers University.

Shieber, Stuart. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8:333–343.

Wartena, Christian. 2000. A note on the complexity of optimality systems. In R. Blutner and G. Jäger, eds., *Studies in Optimality Theory*, pages 64–72. University of Potsdam. Also available at Rutgers Optimality Archive as ROA 385-03100.