Reference-Set Constraints as Linear Tree Transductions via Controlled Optimality Systems

Thomas Graf

Department of Linguistics
University of California, Los Angeles
`tgraf@ucla.edu`
http://tgraf.bol.ucla.edu

Abstract. Reference-set constraints are a special class of constraints used in Minimalist syntax. They extend the notion of well-formedness beyond the level of single trees: When presented with some phrase structure tree, they compute its set of competing output candidates and determine the optimal output(s) according to some economy metric. Doubts have frequently been raised in the literature whether such constraints are computationally tractable [4]. I define a subclass of Optimality Systems (OSs) that is sufficiently powerful to accommodate a wide range of reference-set constraints and show that these OSs are globally optimal [5], a prerequisite for them being computable by linear tree transducers. As regular and linear context-free tree languages are closed under linear tree transductions, this marks an important step towards showing that the expressivity of various syntactic formalisms is not increased by adding reference-set constraints. In the second half of the paper, I demonstrate the feasibility of the OS-approach by exhibiting an efficiently computable OS for a prominent reference-set constraint, Focus Economy [10].

Key words: Optimality Systems, Tree Transducers, Reference-Set Constraints, Transderivationality, Modeling

Introduction

Out of all the items in a syntactician's toolbox, reference-set constraints are probably the most peculiar one. When handed some syntactic tree, a reference-set constraint does not determine its well-formedness from inspection of the tree itself. Instead, it constructs a *reference set* — a set containing a number of trees competing against each other — and chooses the optimal candidate from said set.

Consider *Fewest Steps* [1]. The reference set that this constraint constructs for any given tree t consists of t itself and all the trees that were assembled from the same lexical items as t. All the trees in the reference set are then ranked by the number of movement steps that occurred during their assembly (this is usually identical to the number of traces they contain), and the tree(s) with the fewest instances of movement is (are) chosen as the winner. All other trees are flagged as ungrammatical, including t if it did not emerge as a winner.

Another reference-set constraint is *Focus Economy* [10], which accounts for the empirical fact that neutral stress is compatible with more discourse situations than shifted stress. Take a look at the utterances in (1), where main stress is indicated by **bold face**. Example (1a) can serve as an answer to various questions, among others "What's going on?" and "What did your neighbor buy?". Yet the virtually identical (1b), in which the main stress falls on the subject rather than the object, is compatible only with the question "Who bought a book?". These contrasts indicate a difference as to which constituents may be *focused*, i.e. can be interpreted as providing new information.

(1) a. My neighbor bought a **book**.
   b. My **neighbor** bought a book.

Focus Economy derives the relevant contrast by stipulating that first, any constituent containing the node carrying the sentential main stress can be focused, and second, in a tree in which stress was shifted from the neutral position, a constituent may be focused only if it cannot be focused in the original tree with unshifted stress. In (1a), the object, the VP and the entire sentence can be focused, since these are the constituents containing the main stress carrier. In (1b), the main stress is contained by the subject and the entire sentence, however, only the former may be focused because focusing of the the latter is already a licit option in the neutral stress counterpart (1a).

This esoteric behavior of reference-set constraints coupled with a distinct lack of formal work on their properties has led to various conjectures that they are computationally intractable [4]. In this paper, I refute these claims by showing how reference-set constraints can be emulated by a new variant of Optimality Systems (OSs), and I contend that this route paves the way for reference-set constraints to be implemented as finite-state devices; linear bottom-up tree transducers (lbutts), to be precise. Lbutts are of interest for theoretical as well as practical purposes because both regular and linear context-free tree languages are known to be closed under linear transductions, so applying a linear transducer to a regular/linear context-free tree language yields a regular/linear context-free tree language again. On a theoretical level, this provides us with new insights into the nature of reference-set constraints, while on a practical level, it ensures that adding reference-set constraints to a grammar does not jeopardize its computability. I support my claim by exhibiting a formal model of Focus Economy as an lbutt. My results shed new light on reference-set computation as well as on Optimality Systems and should be of interest to readers from various formal backgrounds, foremost computational phonology and Minimalist grammars.

The paper is laid out as follows: After the preliminaries section, which due to space restrictions has to be shorter than is befitting, I give a brief introduction to OSs before introducing my own variant, controlled OSs, in Sec. 3. The mathematical core results of this section are a new characterization of the important property of global optimality and a simplification of Jäger's theorem [5] regarding the properties of an OS that jointly ensure that it does not exceed the power of linear tree transducers. In the last section, I show how to model Focus Economy as such a restricted OS.

1 Preliminaries and Notation

Let me introduce some notational conventions first. Given a relation R, its *domain* is denoted by $\mathrm{dom}(R)$, its *range* by $\mathrm{ran}(R)$. For any $a \in \mathrm{dom}(R)$, we let $aR := \{b \mid \langle a,b\rangle \in R\}$, unless R is a function, in which case $aR = R(a)$. The *composition* of two relations R and S is $R \circ S := \{\langle a,c\rangle \mid \langle a,b\rangle \in R, \langle b,c\rangle \in S\}$. The *diagonal* of some set A is $id(A) := \{\langle a,a\rangle \mid a \in A\}$.

Tree languages and tree transductions form an integral part of this paper, however, the technical machinery is mostly hidden behind the optimality-theoretic front-end so that only a cursory familiarity with the subject matter is required. Nevertheless the reader is advised to consult [3] and [7] for further details. I also assume that the reader is knowledgeable about string languages and generalized sequential machines.

Definition 1. *A* context-free tree grammar *(CFTG) is defined to be a 4-tuple $\mathcal{G} := \langle \Sigma, F, S, \Delta\rangle$, where Σ and F are disjoint, finite, ranked alphabets of terminals and non-terminals, respectively, and $S \in F$ is the start symbol. Furthermore, Δ is a finite set of productions of the form $F(x_1,\dots,x_n) \to t$, where F is of rank n, and t is a tree with the node labels drawn from $\Sigma \cup F \cup \{x_1,\dots,x_n\}$.*

A production is linear if each variable in its left-hand side occurs at most once in its right-hand side. A CFTG is *linear* if each production is linear. A CFTG is a *regular* tree grammar (RTG) if all non-terminals are of rank 0. A tree language is *regular* iff it is generated by an RTG, and every regular tree language has a context-free language as its string yield.

Definition 2. *A* bottom-up tree transducer *is a 5-tuple $\mathcal{A} := \langle \Sigma, \Omega, Q, Q', \Delta\rangle$, where Σ and Ω are finite ranked alphabets, Q is a finite set of states, and $Q' \subseteq Q$ the set of final states. By Δ we denote a set of productions of the form $f(q_1(x_1),\dots,q_n(x_n)) \to q(t(x_1,\dots,x_n))$, where $f \in \Sigma$ is of rank n, $q_1,\dots,q_n,q \in Q$, and $t(x_1,\dots,x_n)$ is a tree with the node labels drawn from $\Omega \cup \{x_1,\dots,x_n\}$.*

Definition 3. *A* top-down tree transducer *is 5-tuple $\mathcal{A} := \langle \Sigma, \Omega, Q, Q', \Delta\rangle$, where Σ, Ω and Q are as before, $Q' \subseteq Q$ is the set of initial states, and all productions in Δ are of the form $q(f(x_1,\dots,x_n)) \to t$, where $f \in \Sigma$ is of rank n, $q \in Q$ and t is a tree with the node labels drawn from $\Omega \cup \{q(x) \mid q \in Q, x \in \{x_1,\dots,x_n\}\}$.*

As with CFTGs, a production is linear if each variable in its left-hand side occurs at most once in its right-hand side. A transducer is *linear* if each production is linear. I denote a linear bottom-up/top-down tree transducer by lbutt/ltdtt. The class of ltdtts is properly contained in the class of lbutts, which in turn is closed under union and composition. The domain and the range of an lbutt are both recognizable, i.e. regular tree languages. The relation τ induced by a (linear) tree transducer is called a (linear) *tree transduction*. For a bottom-up tree transducer, the graph of τ consists of pairs $\langle s,t\rangle$ such that s and t are Σ- and Ω-labeled trees, respectively, and for some $q \in Q'$, $q(t)$ can be obtained from s by finitely many applications of productions $\delta \in \Delta$. The definition is almost unchanged for top-down tree transducers, except that we require that t can be obtained from $q(s)$. In a slight abuse of terminology, I call a relation *rational* iff it is a finite-state string transduction or a linear tree transduction. For any recognizable tree language L, $id(A)$ is a rational relation. Furthermore, both regular string/tree languages and linear context-free tree languages are closed under rational relations.

In Sec. 4.2, I make good use of $\mathcal{L}^2_{K,P}$ [11], an incarnation of monadic second-order logic (MSO) specifically designed for linguistic purposes. MSO is the extension of first-order logic with monadic second-order variables and predicates as well as quantification over them such that the first-order variables represent nodes in the tree and the second-order variables and predicates sets of nodes. A set of finite strings/trees is definable in MSO iff it is regular. Specifics of $\mathcal{L}^2_{K,P}$ will be briefly introduced in the relevant section. See [11] for further background.

2 Traditional Perspective on Optimality Systems

OSs were introduced independently by [2] and [6] as a formalization of Optimality Theory (OT). In OT, well-formed expressions are no longer derived from underlying representations through iterated applications of string rewrite rules, as was the case with SPE. Instead, underlying representations — which are usually referred to as *inputs* — are assigned a set of *output candidates* by a relation called *generator*, abbreviated Gen. This set is subsequently narrowed down by a sequence of constraints $c_1, \dots, c_n$ until only the *optimal* output candidates remain. This narrowing-down process proceeds in a fashion such that only the candidates that incurred the least number of violations of constraint c_i are taken into account for the evaluation of c_{i+1}. Thus every constraint acts as a (violable) filter on the set of output candidates, with the important addendum that the order in which the filters are applied is crucial in determining optimality.

Consider the example in Fig. 1, which depicts an OT evaluation of output candidates using the tableau notation. Here some input i is assigned three output candidates o_1, o_2 and o_3. The OT grammar uses only three constraints c_1, c_2 and c_3. Constraint c_1 is applied first. Candidates o_2 and o_3 each violate it once, however, o_1 violates it twice. Thus o_2 and o_3 are the output candidates incurring the least number of violations of the constraint and are allowed to proceed to the next round of the evaluation. Candidate o_1, on the other hand, is ruled out and does not participate in further evaluations. Neither o_2 nor o_3 violate c_2 (nor does o_1, but this is immaterial since it has previously been discarded), so neither is filtered out. In the third round, o_2 and o_3 are evaluated with respect to c_3. Each of them violates the constraint once, but since there is no candidate that fares better than them (again, o_1 is not taken into consideration anymore), they also survive this round of the evaluation. Thus, o_2 and o_3 are the optimal output candidates for i. If c_3 had been applied before c_1, on the other hand, o_2 and o_3 would lose out against o_1.

With this intuitive understanding of OT grammars under our belt, the formal definitions of OSs and their *output language* (not to be confused with the *candidate language* ran(Gen)) are straight-forward.

i	c_1	c_2	c_3
o_1	2	0	0
o_2	1	0	1
o_3	1	0	1

Fig. 1. Example of an OT evaluation in tableau notation

Definition 4. *An* optimality system *over languages L and L' is a pair $\mathcal{O} := \langle \textsc{Gen}, C\rangle$ with $\textsc{Gen} \subseteq L \times L'$ and $C := \langle c_1, \ldots, c_n\rangle$ a linearly ordered sequence of functions $c_i \colon \textsc{Gen} \to \mathbb{N}$. For $a, b \in \textsc{Gen}$, $a <_{\mathcal{O}} b$ iff there is a $1 \leq k \leq n$ such that $c_k(a) < c_k(b)$ and for all $j < k$, $c_j(a) = c_j(b)$.*

Definition 5. *Given an optimality system $\mathcal{O} := \langle \textsc{Gen}, C\rangle$, $\langle i, o\rangle$ is* optimal *with respect to $\mathcal{O}$ iff both $\langle i, o\rangle \in \textsc{Gen}$ and there is no o' such that $\langle i, o'\rangle \in \textsc{Gen}$ and $\langle i, o'\rangle <_{\mathcal{O}} \langle i, o\rangle$. The output language of $\mathcal{O}$ is $\mathrm{ran}(\{\langle i, o\rangle \mid \langle i, o\rangle$ is optimal with respect to $\mathcal{O}\})$.*

The important insight of [2] as well as [6], which was later improved upon by [5, 7, 13], is that an OS as defined above can be understood to define a transduction from a set of inputs to its set of optimal output candidates. Moreover, if the OS is suitably restricted, it is guaranteed to define a rational relation, which implies its efficient computability.

Theorem 6. *Let $\mathcal{O} := \langle \textsc{Gen}, C\rangle$ be an OS such that*

- $\mathrm{dom}(\textsc{Gen})$ *is a regular string language, or a regular/linear context-free tree language, and*
- $\textsc{Gen}$ *is a rational relation, and*
- *all $c \in C$ are output-markedness constraints, and*
- *each $c \in C$ defines a rational relation on* $\mathrm{ran}(\textsc{Gen})$*, and*
- *$\mathcal{O}$ is globally optimal.*

Then the transduction τ induced by the OS is a rational relation and $\mathrm{ran}(\tau)$ *belongs to the same formal language class as* $\mathrm{dom}(\tau)$*.*

As the reader might have guessed, the use of τ here is carried over straightforwardly from tree transducers, meaning that $\tau := \{\langle i, o\rangle \mid \langle i, o\rangle$ is optimal with respect to $\mathcal{O}\}$. The theorem also makes reference to two notions we have not encountered yet at all, output-markedness and global optimality. The former is easily defined.

Definition 7. *Given an OS $\mathcal{O} := \langle \textsc{Gen}, C\rangle$, $c \in C$ is an* output-markedness constraint *iff $c(\langle i, o\rangle) = c(\langle i', o\rangle)$ for all $\langle i, o\rangle, \langle i', o\rangle \in \textsc{Gen}$.*

Global optimality, on the other hand, requires a lot of finite-state machinery to be in place before it can be made formally precise, which would lead us off track here. For our purposes it is sufficient to know that an OS is globally optimal iff for every optimal output candidate o it holds that there is no input i such that o is an output candidate for i but not an optimal one. The curious reader is referred to [5] for a more rigorous definition.

3 Controlled Optimality Systems

OSs are perfectly capable of modeling reference-set constraints. The reference set for any input i is defined by iGen, and the evaluation metric can straightforwardly be implemented as a sequence of constraints. Fewest Steps, for instance, can be viewed as an OS in which a tree t is related by Gen to all the trees that were constructed from the same lexical items as t, including t itself. Besides that, the OS has only one constraint *Trace, which punishes traces. As a consequence, only the trees with the least number of traces will be preserved, and these are the optimal output candidates for t. While this short example shows that OSs can get the job done, the way output candidates are specified for reference-set constraints actually relies on additional structure — the reference sets — that is only indirectly represented by Gen. In the following I introduce controlled OSs as a variant of standard OSs that is closer to reference-set computation by making reference sets prime citizens of OSs and demoting Gen to an ancillary relation that is directly computed from them.

We observe first that many reference-set constraints allow for distinct inputs to be assigned the same reference set. Hence it makes sense to map entire sets of inputs to reference sets, rather than the individual inputs themselves. Let us call such a set of inputs a *reference type*. An OS can then be defined by reference types and a function mapping them to reference sets:

Definition 8 (Controlled Optimality Systems). *An* $\mathcal{F}$-controlled optimality system *over languages* L, L' *is a 4-tuple* $\mathcal{O}[\mathcal{F}] := \langle \text{Gen}, C, \mathcal{F}, \gamma \rangle$, *where*

- Gen *and* C *are defined as usual,*
- $\mathcal{F}$ *is a family of non-empty subsets of* L, *each of which we call a* reference type,
- *the* control map $\gamma : \mathcal{F} \to \wp(L') \setminus \{\emptyset\}$ *associates every reference type with a* reference set, *i.e. a set of output candidates,*
- *the following conditions are satisfied*
  - exhaustivity*:* $\bigcup_{X \in \mathcal{F}} X = L$
  - bootstrapping*:* $x\text{Gen} = \bigcup_{x \in X \in \mathcal{F}} X\gamma$

Every controlled OS can be translated into a canonical OS by discarding its third and fourth component (i.e. $\mathcal{F}$ and the control map γ). In the other direction, a controlled OS can be obtained from every canonical OS by setting $\mathcal{F} := \{\{i\} \mid i \in L\}$ and $\gamma : \{i\} \mapsto i\text{Gen}$. So the only difference between the two is that controlled OSs modularize Gen by specifying it through reference types and the control map.

Now that OSs operate (at least partially) at the level of sets, it will often be interesting to talk about the set of optimal output candidates assigned to some reference type, rather than one particular input. But whereas the set of optimal output candidates is always well-defined for inputs — for any input i this set is given by $i\tau$ — we have to be more careful when lifting it to reference types, because distinct inputs that belong to the same reference type may not necessarily be assigned the same optimal output candidates. Such a situation might arise in OSs with faithfulness or input-markedness constraints, which are sensitive to properties of the input, or when two inputs i and j are of reference type X, but in addition j is also of reference type Y. Given this ambiguity, one has to distinguish between the set of output candidates that are optimal for at least one member of reference type X, and the set of output candidates that are optimal for all members of reference type X. The former is the *up-transduction* $X\tau^{\uparrow} := \bigcup_{x \in X} x\tau$, the latter the *down-transduction* $X\tau^{\downarrow} := \bigcap_{x \in X} x\tau$.

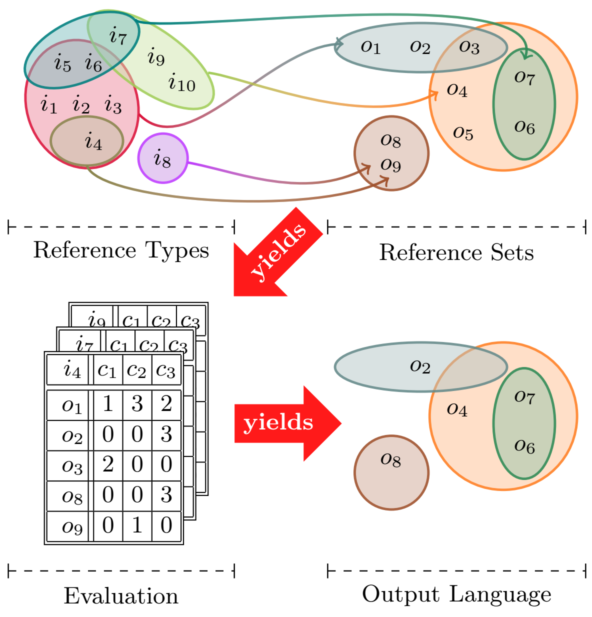
At this point it might be worthwhile to work through a simple example. Fig. 2 on the following page depicts a controlled OS and the distinct steps of its computation. We are given a collection of reference types consisting of $\textsc{Red} := \{i_1, i_2, i_3, i_4, i_5, i_6\}$, $\textsc{Sienna} := \{i_4\}$, $\textsc{Teal} := \{i_5, i_6, i_7\}$, $\textsc{Purple} := \{i_8\}$, and $\textsc{Lime} := \{i_7, i_9, i_{10}\}$. The reference sets are $\textsc{Blue} := \{o_1, o_2, o_3\}$, $\textsc{Orange} := \{o_3, o_4, o_5, o_6, o_7\}$, $\textsc{Green} := \{o_6, o_7\}$, and $\textsc{Brown} := \{o_8, o_9\}$. Finally, the graph of γ consists of the pairs $\langle \textsc{Red}, \textsc{Blue} \rangle$, $\langle \textsc{Sienna}, \textsc{Brown} \rangle$, $\langle \textsc{Teal}, \textsc{Green} \rangle$, $\langle \textsc{Purple}, \textsc{Brown} \rangle$, and $\langle \textsc{Lime}, \textsc{Orange} \rangle$. Note that a reference type may overlap with another reference type or even be a proper subset of it, and the same holds for reference sets. This means that an input can belong to several reference types at once. Consequently, $x\textsc{Gen}$ may be a superset of $X\gamma$ for every reference type X that contains x, as is the case for i_4, say, but not for i_7, even though both are assigned exactly two reference types. Input i_4 is related by $\textsc{Gen}$ to all outputs contained in $\textsc{Red}\gamma \cup \textsc{Sienna}\gamma = \textsc{Blue} \cup \textsc{Brown} = \{o_1, o_2, o_3, o_8, o_9\}$, whereas i_7 is related to $\textsc{Lime}\gamma \cup \textsc{Teal}\gamma = \textsc{Orange} \cup \textsc{Green} = \textsc{Orange} = \{o_3, o_4, o_5, o_6, o_7\}$. As soon as $\textsc{Gen}$ has been determined from the reference types and the control map, the computation proceeds as usual with the constraints of the OS filtering out all suboptimal candidates.

Interestingly, almost all reference-set constraints fall into two classes with respect to how reference types and reference sets are distributed. In the case of Fewest Steps, where the input language is also the candidate language, each reference type is mapped to itself, that is to say, there is no distinction between reference types and reference sets. A constraint like Focus Economy, on the other hand, requires not only the input language and the candidate language to be disjoint, but also all reference sets and reference types. A natural unification of these two subclasses is available in the form of *output joint preservation.*

Definition 9. *An $\mathcal{F}$-controlled optimality system is* output joint preserving *iff for all distinct $X, Y \in \mathcal{F}$, $X\gamma \cap Y\gamma \neq \emptyset \rightarrow X \cap Y \neq \emptyset$.*

The OS depicted in Fig. 2 on the next page fails output joint preservation. It is clearly violated by $\textsc{Sienna}$ and $\textsc{Purple}$, which are disjoint yet mapped to the same reference set, $\textsc{Brown}$. It isn't respected by $\textsc{Red}$ and $\textsc{Lime}$, either, which are mapped to $\textsc{Blue}$ and $\textsc{Orange}$, respectively, the intersection of which is non-empty even though $\textsc{Red}$ and $\textsc{Lime}$ are disjoint.

Output joint preservation is certainly general enough a property to encompass the kinds of controlled OSs we are interested in. In the following, I show that it is also sufficiently restrictive to establish a link to the crucial notion of global optimality.



i_4	c_1	c_2	c_3
o_1	1	3	2
o_2	0	0	3
o_3	2	0	0
o_8	0	0	3
o_9	0	1	0

Fig. 2. Example of a controlled OS. Gen is defined in a modular fashion using reference types, reference sets, and the control map γ from reference types to reference sets.

Definition 10. *An $\mathcal{F}$-controlled OS is* type-level optimal *iff* $X\tau^{\uparrow} \restriction X\gamma = X\tau^{\downarrow} \restriction X\gamma$ *for all* $X \in \mathcal{F}$.

Lemma 11. *Let $\mathcal{O}[\mathcal{F}]$ an $\mathcal{F}$-controlled OS. Then $\mathcal{O}[\mathcal{F}]$ is type-level optimal only if it is globally optimal.*

Proof. We prove the contrapositive. If $\mathcal{O}[\mathcal{F}]$ is not type-level optimal, then it holds for some $X \in \mathcal{F}$ that $X\tau^{\uparrow} \restriction X\gamma \neq X\tau^{\downarrow} \restriction X\gamma$. But this implies that there are $x, y \in X$ and $z \in X\gamma$ such that $x\tau \ni z \notin y\tau$, which is an unequivocal violation of global optimality. □

Theorem 12. *Every output joint preserving OS is type-level optimal iff it is globally optimal.*

Proof. The right-to-left direction follows from Lemma 11. We prove the contrapositive of the other direction. If $\mathcal{O}[\mathcal{F}]$ fails global optimality, then there are $x, y \in L$ and $z \in L'$ such that $\langle x, z\rangle, \langle y, z\rangle \in$ Gen yet $x\tau \ni z \notin y\tau$. W.l.o.g. let $x \in X$ and $y \in Y$, $X, Y \in \mathcal{F}$, whence $z \in X\gamma \cap Y\gamma$. As $\mathcal{O}[\mathcal{F}]$ is output joint preserving, $X\gamma \cap Y\gamma \neq \emptyset$ entails $X \cap Y \neq \emptyset$. Pick some $p \in X \cap Y$. Now if $\mathcal{O}[\mathcal{F}]$ is type-level optimal, then it holds that $X\tau^{\uparrow} \restriction X\gamma = X\tau^{\downarrow} \restriction X\gamma$ and $Y\tau^{\uparrow} \restriction Y\gamma = Y\tau^{\downarrow} \restriction Y\gamma$, so $z \in x\tau$ implies $z \in p\tau$, whereas $z \notin y\tau$ implies $z \notin p\tau$. Contradiction. It follows that $\mathcal{O}[\mathcal{F}]$ is not type-level optimal. □

Intuitively, type-level optimality ensures that optimality is fixed for entire reference types, so the individual inputs can be ignored for determining optimality. However, it is too weak a restriction to rule out disagreement between reference types that are mapped to overlapping reference sets, so output joint preservation has to step in; it guarantees that if two reference types X and Y share at least one output candidates, there exists some input p belonging to both X and Y that will be faced by conflicting requirements if X and Y disagree with respect to which candidates in $X\gamma \cap Y\gamma$ they deem optimal (since the OS is type-level optimal, optimality can be specified for entire reference types rather than their members). It should be easy to see that the conditions jointly imply global optimality.

Given our interest in using controlled OS to investigate the computability of reference-set constraints, it would be advantageous if we could read off the constraints right away whether they yield type-level optimality. This is indeed very easy to do thanks to the following entailment.

Lemma 13. *Let $\mathcal{O}[\mathcal{F}] := \langle \textsc{Gen}, C, \mathcal{F}, \gamma \rangle$ an $\mathcal{F}$-controlled OS such that every $c \in C$ is an output-markedness constraint. Then $\mathcal{O}[\mathcal{F}]$ is type-level optimal.*

Proof. Assume the opposite. Then for some $X \in \mathcal{F}$, $X\tau^{\uparrow} \restriction X\gamma \neq X\tau^{\downarrow} \restriction X\gamma$, whence there are $x, y \in X$ and $z \in X\gamma$ with $x\tau \ni z \notin y\tau$. But this is the case only if there is some $c \in C$ such that $c(\langle x, z \rangle) \neq c(\langle y, z \rangle)$, i.e. c isn't an output-markedness constraint. $\square$

Corollary 14. *Let $\mathcal{O}[\mathcal{F}] := \langle \textsc{Gen}, C, \mathcal{F}, \gamma \rangle$ an output joint preserving OS such that every $c \in C$ is an output-markedness constraint. Then $\mathcal{O}[\mathcal{F}]$ is globally optimal.*

Combining these results, we arrive at the equivalent of Thm. 6 for $\mathcal{F}$-controlled OSs.

Corollary 15. *Let $\mathcal{O}[\mathcal{F}] := \langle \textsc{Gen}, C, \mathcal{F}, \gamma \rangle$ an $\mathcal{F}$-controlled OS such that*

- dom(Gen) *is a regular string language, or a regular/linear context-free tree language, and*
- Gen *is a rational relation, and*
- *all $c \in C$ are output-markedness constraints, and*
- *each $c \in C$ defines a rational relation on* ran(Gen)*, and*
- *$\mathcal{O}[\mathcal{F}]$ is output joint preserving.*

Then the transduction τ induced by the OS is a rational relation and $\mathrm{ran}(\tau)$ *belongs to the same formal language class as* $\mathrm{dom}(\tau)$*.*

In sum, then, not only do output joint preserving OSs look like a solid base for modeling reference-set constraints, they also have the neat property that the global optimality check is redundant, thanks to Lem. 13. As it is pretty easy to determine for any given reference-set constraint whether it can be modeled by output-markedness constraints alone, the decisive factor in their implementation are the transducers for the constraints and Gen. If those transducers each define a rational relation, so does the entire optimality system.

4 Application to Reference-Set Computation

4.1 Focus Economy Explained

Focus Economy [10] was briefly discussed in the introduction. It is invoked in order to account for the fact that sentences such as (2a), (2b) and (2c) below differ with respect to what is given and what is new information. Once again main stress is marked by **boldface**.

(2) a. My friend bought a red **car**.
b. My friend **bought** a red car.
c. My friend bought a **red** car.

That these utterances are associated to different information structures is witnessed by the fact that for instance only (2a) is compatible with the question "What happened?".

The full-blown Focus Economy system (rather than the simplified sketch given in the introduction) accounts for the data as follows. First, the *Main Stress Rule* demands that in every pair of sister nodes, the "syntactically more embedded" node [10, p.133] is assigned strong stress, its sister weak stress (marked in the phrase structure tree by subscripted S and W, respectively). If a node has no sister, it is always assigned strong stress (in Minimalist syntax, this will be the case only for the root node, as all Minimalist trees are strictly binary branching). Main stress then falls on the unique leaf node that is connected to the root node by a path of nodes that have an S-subscript. See Fig. 3 for an example.

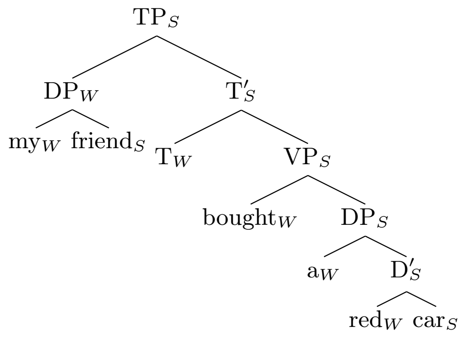


Fig. 3. The stress-annotated phrase structure tree for (2a).

The notion of being syntactically more embedded isn't explicitly defined in the literature. It is stated in passing, though, that "... main stress falls on the most embedded constituent on the recursive side of the tree" [10, p.133]. While this is rather vague, it is presumably meant to convey that, at least for English, in which complements follow the heads they are introduced by, the right sister node is assigned strong stress as long as it isn't an adjunct. This interpretation seems to be in line with the empirical facts.

The second integral part of the proposal is the operation *Stress Shift*, which shifts the main stress to some leaf node n by assigning all nodes on the path from n to the root strong stress and demoting the sisters of these nodes to weakly stressed nodes. For instance, the tree for "**My** friend bought a red car" is obtained from the tree in Fig. 3 by changing my_W and friend_S to my_S and friend_W, respectively, and DP_W and T'_S to DP_S and T'_W, respectively.

While Stress Shift could be invoked to move stress from anaphoric elements to their left sister as in (3), this burden is put on a separate rule, for independent reasons. The rule in question is called *Anaphoric Destressing* and obligatorily assigns weak stress to anaphoric nodes, where a node is anaphoric iff it is "... D[iscourse]-linked to an accessible discourse entity" [10, p.147]. Thus Anaphoric Destressing not only accounts for the unstressed anaphor in (3), but also for the special behavior of stress in cases of parallelism.

(3) John **killed** her.

(4) First Paul bought a red **car**.
  a. Then **John** bought one.
  b. * Then John bought **one**.

The overall system now works as follows. Given a phrase structure tree that has not been annotated for stress yet, one first applies Anaphoric Destressing to make sure that all d-linked constituents are assigned weak stress and thus cannot carry main stress. Next the Main Stress Rule is invoked to assign every node in the tree either W or S. Note that the Main Stress Rule cannot overwrite previously assigned labels, so if some node n has been labeled W by Anaphoric Destressing, the Main Stress Rule has to assign S to the sister of n. Now that the tree is fully annotated, we compute its *focus set*, the set of constituents that may be focused.

(5) *Focus Projection*
Given some stress-annotated tree t, its focus set is the set of nodes reflexively dominating the node carrying main stress.

The focus set of "My friend bought a red **car**", for instance, contains [car], [.D′ red car], [.DP a red car], [.VP bought a red car] and [.TP My friend bought a red car]. For "Then **John** bought one", on the other hand, it consists only of [John] and [.TP Then John bought one].

At this point, Stress Shift may optionally take place. After the main stress has been shifted, however, the focus set has to be computed all over again, and this time the procedure involves reference-set computation.

(6) *Focus Projection Redux*
Given some stress-annotated tree t' that was obtained from tree t by Stress Shift, the focus set of t' contains all the nodes reflexively dominating the node carrying main stress which aren't already contained in the focus set of t.

So if "Then **John** bought one" had been obtained by Stress Shift from [.TP Then John bought one] rather than Anaphoric Destressing, its focus set would have contained only [John], because [.TP Then John bought one] already belongs to the focus set of "Then John bought **one**". As an easy exercise, the reader may want to draw annotated trees for the examples in (2) and compute their focus sets.

4.2 A Model of Focus Economy

After this general overview, the time has come to formalize Focus Economy. In order to precisely model Focus Economy, though, I have to make some simplifying assumptions, for reasons that are entirely independent from the restrictions of OSs. First, I stipulate that adjuncts are explicitly marked as such by a subscript A on their label. This is simply a matter of convenience, as it reduces the complexity of the transducers and makes my model independent from the theoretical status of adjuncts in syntax.

Second, I decided to take movement out of the picture, because the interaction of focus and movement is not touched upon in [10], so there is no original material to formalize. Incidentally, movement seems to introduce several interesting complications (e.g. mandatory main stress for topicalized constituents). The end of this section contains a brief discussion as to whether my model can be extended to capture theories involving movement.

The last simplification concerns Anaphoric Destressing. While the destressing of pronominal (and possibly covert) elements is easy to accommodate, the invoked notion of d-linking is impossible to capture in any model that operates on isolated syntactic trees. Devising a working model of discourse structure vastly exceeds the scope of this contribution. Also, the role of d-linking in anaphoric destressing is of little importance to this paper, which focuses on the reference-set computational aspects of Focus Economy. Thus my implementation will allow almost any constituent to be anaphorically distressed and leave the task of matching trees to appropriate discourse contexts to an external theory of d-linking that remains to be specified.

With these provisions made explicit, the formalization of Focus Economy as a controlled OS can commence. The input language is supposedly derived by some movement-free Minimalist grammar $\mathcal{E}$ for English [12] in which interior nodes are given explicit category labels (once more for the sake of convenience). As Minimalist grammars without remnant movement generate regular tree languages [8], it is safe to assume that Minimalist grammars without any kind of movement do so, too.

Next I define GEN as the composition of four linear transducers corresponding to Anaphoric Distressing, the Main Stress Rule, Stress Shift, and Focus Projection, respectively. Given a tree t derived by $\mathcal{E}$, the transducer cascade computes all logically possible variants of t with respect to stress assignment and then computes the focus in a local way. This means that GEN actually overgenerates with respect to focus, a problem that we have to take care of at a letter step.

Anaphoric Distressing is modeled by a non-deterministic ltdtt that may randomly add a subscript D to a node's label in order to mark it as anaphoric. The only condition is that if a node is labeled as anaphoric, all the nodes it properly dominates must be marked as such, too.

Definition 16. *Let $\Sigma := \Sigma_L \cup \Sigma_A$ be the vocabulary of the Minimalist grammar $\mathcal{E}$ that generated the input language, where Σ_L contains all lexical items and category labels and Σ_A their counterparts explicitly labeled as adjuncts.* Anaphoric Destressing *is the ltdtt $\mathcal{D}$ where $\Sigma_{\mathcal{D}} := \Sigma$, $\Omega_{\mathcal{D}}$ is the union of Σ and $\Sigma_D := \{\sigma_D \mid \sigma \in \Sigma\}$, $Q := \{q_i, q_d\}$, $Q' := \{q_i\}$, and $\Delta_{\mathcal{D}}$ contains the rules below, with $\sigma \in \Sigma$ and $\sigma_D \in \Sigma_D$ and $\alpha_{\{x,y\}}$ to be read as "α_x or α_y":*

$$\begin{array}{rr} q_i(\sigma(x,y)) \to \sigma(q_i(x), q_i(y)) & q_i(\sigma) \to \sigma \\ q_{\{i,d\}}(\sigma(x,y)) \to \sigma_D(q_d(x), q_d(y)) & q_{\{i,d\}}(\sigma) \to \sigma_D \end{array}$$

The transducer for the Main Stress Rule is non-deterministic, too, but it proceeds in a bottom-up manner. It does not alter nodes subscripted by A or D, but if it encounters a leaf node without a subscript, it randomly adds the subscript S or W to its label. However, W is allowed to occur only on left sisters, whereas S is mostly restricted to right sisters and may surface on a left sister just in case the right sister is already marked by A or D. Note that we could easily define a different stress pattern, maybe even parametrized with respect to category labels, to incorporate stress assignment rules from other languages.

Definition 17. Main Stress *is the lbutt $\mathcal{M}$ where $\Sigma_{\mathcal{M}} := \Omega_{\mathcal{D}}$, $\Omega_{\mathcal{M}}$ is the union of Σ, Σ_D and $\Sigma_* := \{\sigma_S, \sigma_W \mid \sigma \in \Sigma\}$, $Q := \{q_s, q_u, q_w\}$, $Q' := \{q_s\}$ and $\Delta_{\mathcal{M}}$ contains the following rules, with $\sigma \in \Sigma$, $\sigma_A \in \Sigma_A$, $\sigma_x \in \{\sigma_x \mid \sigma \in \Sigma\}$ for $x \in \{D, S, W\}$, and $\alpha_{a\ldots z}(\beta_{a'\ldots z'}, \ldots, \zeta_{a'',\ldots,z''})$ to be read as "$\alpha_a(\beta_{a'}, \ldots, \zeta_{a''})$ or $\ldots$ or $\alpha_z(\beta_{z'}, \ldots, \zeta_{z''})$":*

$$\begin{array}{rr} \sigma_A \to q_u(\sigma_A) & \sigma_A(q_u(x), q_u(y)) \to q_u(\sigma_A(x,y)) \\ \sigma_D \to q_u(\sigma_D) & \sigma_D(q_u(x), q_u(y)) \to q_u(\sigma_D(x,y)) \\ \sigma \to q_{sw}(\sigma_{SW}) & \sigma(q_{\{u,w\}}(x), q_s(y)) \to q_{sw}(\sigma_{SW}(x,y)) \\ & \sigma(q_s(x), q_u(y)) \to q_{sw}(\sigma_{SW}(x,y)) \end{array}$$

Stress Shift is best implemented as a non-deterministic ltdtt that may randomly switch the subscripts of two S/W-annotated sisters.

Definition 18. Stress Shift *is the ltdtt $\mathcal{S}$ where $\Sigma_{\mathcal{S}} = \Omega_{\mathcal{S}} = \Omega_{\mathcal{M}}$, $Q := \{q_i, q_s, q_w\}$, $Q' := \{q_s\}$, and $\Delta_{\mathcal{S}}$ contains the rules below, with $\sigma \in \Sigma_{\mathcal{S}}$ and $\sigma_* \in \Sigma_*$:*

$$\begin{array}{rr} q_s(\sigma_*(x,y)) \to \sigma_{SSS}(q_{isw}(x), q_{iws}(y)) & q_s(\sigma_*) \to \sigma_S \\ q_w(\sigma_*(x,y)) \to \sigma_W(q_i(x), q_i(y)) & q_w(\sigma_*) \to \sigma_W \\ q_i(\sigma(x,y)) \to \sigma(q_i(x), q_i(y)) & q_i(\sigma) \to \sigma \end{array}$$

The last component is Focus Projection, a non-deterministic ltdtt with two states, q_f and q_g. The transducer starts at the root in q_f. Whenever a node

n is subscripted by W, the transducer switches into q_g at this node and stays in the state for all nodes dominated by n. As long as the transducer is in q_f, it may randomly add a superscript F to a label to indicate that it is focused. Right afterward, it changes into q_g and never leaves this state again. Rather than associating a stress-annotated tree with a set of constituents that can be focused, Focus Projection now generates multiple trees that differ only with respect to which constituent along the path of S-labeled nodes is focus-marked.

Definition 19. Focus Projection *is the ltdtt $\mathcal{F}$, where $\Sigma_{\mathcal{F}} = \Omega_{\mathcal{S}}$, $\Omega_{\mathcal{F}}$ is the union of $\Omega_{\mathcal{S}}$ and $\Omega_{\mathcal{S}}^F := \{\omega^F \mid \omega \in \Omega_{\mathcal{S}}\}$, $Q := \{q_f, q_g\}$, $Q' := \{q_f\}$, and $\Delta_{\mathcal{F}}$ contains the rules below, with $\sigma \in \Sigma_{\mathcal{F}}$ and $\sigma_{\overline{S}} \in \Sigma_{\mathcal{F}} \setminus \{\sigma_S \mid \sigma \in \Sigma\}$:*

$$\begin{aligned}
q_f(\sigma_S(x,y)) &\to \sigma_S(q_f(x), q_f(y)) & & \\
q_f(\sigma_S(x,y)) &\to \sigma_S^F(q_g(x), q_g(x)) & q_f(\sigma_S) &\to \sigma_S^F \\
q_f(\sigma_{\overline{S}}(x,y)) &\to \sigma_{\overline{S}}(q_g(x), q_g(x)) & q_f(\sigma_{\overline{S}}) &\to \sigma_{\overline{S}} \\
q_g(\sigma(x,y)) &\to \sigma(q_g(x), q_g(y)) & q_g(\sigma) &\to \sigma
\end{aligned}$$

All four transducers are linear, whence they can be composed into a single linear transducer modeling Gen. Expanding on what was said above about the inner workings of Gen, we now see that for any tree t in the input language, tGen is the set of stress-annotated trees in which, first, some subtrees may be marked as adjuncts or anaphorical material (or both) and thus do not carry stress information, second, there is exactly one path from the root to some leaf such that every node in the path is labeled by S, and third, exactly one node belonging to this path is marked as focused. The reader should have no problem verifying that in terms of controlled OSs, all reference types are singleton and their reference-sets do not overlap, i.e. output joint preservation and type-level optimality are satisfied.

Now it only remains for us to implement *Focus Projection Redux.* In the original account, Focus Projection Redux applied directly to the output of Stress Shift, i.e. trees without focus information, and the task at hand was to assign the correct focus. In my system, on the other hand, every tree is fed into Focus Projection and marked accordingly for focus. This leads to overgeneration for trees in which Stress Shift has taken place — a node may carry focus even if it could also do so in the tree without shifted main stress. Consequently, the focus set of "**John** died", for instance, turns out to contain both [John] and [.TP John died] rather than just the former. Under my proposal, then, Focus Projection Redux is faced with the burden of filtering out focus information instead of assigning it. In other words, Focus Projection Redux is a constraint. This is accomplished by defining a regular tree language L_c such that when Gen is composed with the diagonal of L_c (which is guaranteed to be a linear transduction), only trees with licit focus marking are preserved; said regular language is easily specified in the monadic second-order logic $\mathcal{L}^2_{K,P}$ [11].

First one defines two predicates, *StressPath* and *FocusPath*. The former picks out the path from the root to the leaf carrying main stress, whereas the latter refers to the path from the root to the leaf that would carry main stress in the absence of stress shift. This implies that *FocusPath* replicates some of the information that is already encoded in the Main Stress transducer. Note that in the definitions below, $A(x)$, $D(x)$ and $S(x)$ are predicates picking out all nodes with subscript A, D, S, respectively, $x \lhd y$ denotes "x is the parent of y", $x \prec y$ "x is the left sibling of y", and $\lhd^*$ the reflexive transitive closure of $\lhd$.

$$\begin{aligned}&\text{Path}(X) \leftrightarrow \exists x\Big[X(x) \wedge \neg\exists y[y \lhd x]\Big] \wedge \exists! x\Big[X(x) \wedge \neg\exists y[x \lhd y]\Big] \wedge \\ &\forall x, y, z\Big[\big(X(x) \wedge X(y) \rightarrow x \lhd^* y \vee y \lhd^* x\big) \wedge \big(X(x) \wedge \neg X(z) \rightarrow \neg(z \lhd^* x)\big)\Big]\end{aligned}$$

$$\text{StressPath}(X) \leftrightarrow \text{Path}(X) \wedge \forall x[X(x) \rightarrow S(x)]$$

$$\begin{aligned}&\text{FocusPath}(X) \leftrightarrow \text{Path}(X) \wedge \forall x, y, z\Big[X(x) \wedge x \lhd y \wedge x \lhd z \rightarrow \\ &\qquad \big((A(y) \vee D(y)) \rightarrow X(z)\big) \wedge \big(\neg A(y) \wedge \neg D(y) \wedge y \prec z \rightarrow X(z)\big)\Big]\end{aligned}$$

In a tree where no stress shift has taken place, StressPath and FocusPath are true of the same subsets and any node contained by them may be focused. After an application of the Stress Shift rule, however, the two paths are no longer identical, although their intersection is never empty (it has to contain at least the root node). In this case, then, the only valid targets for focus are those nodes of the StressPath that are not contained in the FocusPath. This is formally expressed by the $\mathcal{L}^2_{K,P}$ sentence ϕ below. Just like $A(x)$, $D(x)$ and $S(x)$ before, $F(x)$ is a predicate defining a particular set of nodes, this time the set of nodes labeled by some $\omega^F \in \Omega^F_S$. I furthermore use $X \approx Y$ as a shorthand for $\forall x[X(x) \leftrightarrow Y(x)]$.

$$\phi := \forall x, X, Y[F(x) \wedge X(x) \wedge \text{StressPath}(X) \wedge \text{FocusPath}(Y) \rightarrow (Y(x) \rightarrow X \approx Y)]$$

Note that ϕ by itself does not properly restrict the distribution of focus. First of all, there is no requirement that exactly one node must be focused. Second, nodes outside StressPath may carry focus, in which case no restrictions apply to them at all. Finally, StressPath and FocusPath may be empty, because we have not made any assumptions about the distribution of labels. Crucially, though, ϕ behaves as expected over the trees in the candidate language. Thus taking the diagonal of the language licensed by ϕ and composing it with GEN filters out all illicit foci, and only those. Since the diagonal over a regular language is a linear transduction, the transduction obtained by the composition is too. This establishes the computational feasibility of Focus Economy when the input language is a regular tree language.

So far I have left open the question, though, how movement fits into the picture. First of all, it cannot be ruled out *a priori* that the interaction of movement and focus are so intricate on a linguistic level that significant modifications have to be made to the original version of Focus Economy. On a formal level, this would mean that the transduction itself would have to be changed. In this case, it makes little sense to speculate how my model could be extended to accommodate movement, so let us instead assume that Focus Economy can remain virtually unaltered and it is only the input language that has to be modified. In my model, the input language is a regular tree language by virtue of being generated by a Minimalist grammar without movement. But note that Minimalist grammars with movement generate regular tree languages, too, in the presence of a ban against more exotic kinds of movement such as remnant movement or head movement [8]. Thus the restriction to regular tree languages itself does not preclude us from accommodating most instances of movement.[1]

Conclusion

I have shown that despite claims to the contrary, reference-set constraints aren't computationally intractable. Controlled OSs were introduced as a formal model for reference-set constraints. I gave a new characterization of global optimality for the subclass of output joint preserving OSs, which is general enough to accommodate most reference-set constraints. The shift in perspective induced by controlled OSs made it apparent that out of the five conditions that jointly guarantee for an OS to stay within the limits of linear tree transductions, three are almost trivially satisfied by reference-set constraints, with the only problematic areas being the power of Gen and the rankings induced by the constraints on the range of Gen. A model of a prominent reference-set constraint, Focus Economy, showed that at least for some constraints those point aren't problematic, either. These new results suggest that reference-set constraints are significantly better behaved than is usually believed.

Acknowledgments

I am greatly indebted to Ed Stabler, Ed Keenan and Uwe Mönnich as well as the two anonymous reviewers for their motivational comments and helpful criticism. The research reported herein was supported by a DOC-fellowship of the Austrian Academy of Sciences.

Bibliography

[1] Chomsky, N.: The Minimalist Program. MIT Press, Cambridge, Mass. (1995)

[1] If we want the full expressive power of Minimalist grammars, then the best strategy is to express Focus Economy as a constraint over derivation trees, since for every Minimalist grammar the set of derivation trees it licenses forms a regular language that fully determines the tree yield of the grammar [9]. The only difference between Minimalist derivation trees and movement-free phrase structure trees as derived above is that the latter are unordered. Hence, if we require that linear order (which can be easily determined from the labels of the leaves) is directly reflected in the derivation trees, the formalization above carries over unaltered to derivation trees and may be extended as desired to deal with instances of movement.

[2] Frank, R., Satta, G.: Optimality theory and the generative complexity of constraint violability. Computational Linguistics 24, 307–315 (1998)

[3] Gécseg, F., Steinby, M.: Tree Automata. Academei Kaido, Budapest (1984)

[4] Johnson, D., Lappin, S.: Local Constraints vs. Economy. CSLI, Stanford (1999)

[5] Jäger, G.: Gradient constraints in finite state OT: The unidirectional and the bidirectional case. In: Kaufmann, I., Stiebels, B. (eds.) More than Words. A Festschrift for Dieter Wunderlich, pp. 299–325. Akademie Verlag, Berlin (2002)

[6] Karttunen, L.: The proper treatment of optimality in computational phonology (1998), manuscript, Xerox Research Center Europe

[7] Kepser, S., Mönnich, U.: Closure properties of linear context-free tree languages with an application to optimality theory. Theoretical Computer Science 354, 82–97 (2006)

[8] Kobele, G.M.: Without remnant movement, MGs are context-free. In: Ebert, C., Jäger, G., Michaelis, J. (eds.) MOL 10/11. Lecture Notes in Computer Science, vol. 6149, pp. 160–173 (2010)

[9] Kobele, G.M., Retoré, C., Salvati, S.: An automata-theoretic approach to minimalism. In: Rogers, J., Kepser, S. (eds.) Model Theoretic Syntax at 10. pp. 71–80 (2007), workshop organized as part of the Europen Summer School on Logic, Language and Information, ESSLLI 2007, 6-17 August 2007, Dublin, Ireland

[10] Reinhart, T.: Interface Strategies: Optimal and Costly Computations. MIT Press, Cambridge, Mass. (2006)

[11] Rogers, J.: A Descriptive Approach to Language-Theoretic Complexity. CSLI, Stanford (1998)

[12] Stabler, E.P., Keenan, E.: Structural similarity. Theoretical Computer Science 293, 345–363 (2003)

[13] Wartena, C.: A note on the complexity of optimality systems. In: Blutner, R., Jäger, G. (eds.) Studies in Optimality Theory, pp. 64–72. University of Potsdam, Potsdam, Germany (2000)