UCLA Working Papers in Linguistics, Papers in Mathematical Linguistics 1
Volume 15, Article 4: 1–53, 2010

A Tree Transducer Model of Reference-Set Computation

Thomas Graf

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 (Johnson and Lappin 1999). 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 in the sense of Jäger (2002), 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-based transducer approach by exhibiting the transducers for three prominent reference-set constraints, Focus Economy (Reinhart 2006), Merge-over-Move (Chomsky 1995b), and Fewest Steps (Chomsky 1991, 1995b). My approach sheds new light on the internal mechanics of these constraints and suggests that their advantages over standard well-formedness conditions have not been sufficiently exploited in the empirical literature.

Keywords reference-set constraints, transderivationality, Optimality Systems, tree transducers, modelling, finite-state methods, Focus Economy, Merge-over-Move, Fewest Steps

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*, also known as the *Shortest Derivation Principle* (Chomsky 1991, 1995a). 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, 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* (Szendrői 2001; Reinhart 2006), 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).

The application domain of reference-set constraints includes narrow syntax as well as the interfaces. In syntax, one finds Fewest Steps (Chomsky 1995b), Merge-over-Move (Chomsky 1995b, 2000), Pronouns as Last Resort (Hornstein 2001), resumption in Lebanese Arabic (Aoun, Choueiri, and Hornstein 2001), phrase structure projection (Toivonen 2001), the Person Case Constraint (Rezac 2007), Chain Uniformization (Nunes 2004), object extraction in Bantu (Nakamura 1997), and many others. The most prominent interface constraints are Rule I (Grodzinsky and Reinhart 1993; Heim 1998; Reinhart 2006; Heim 2009), Scope Economy (Fox 1995, 2000), and the previously mentioned Focus Economy, but there are also more recent proposals such as Situation Economy (Keshet 2010).

The somewhat esoteric behavior of reference-set constraints coupled with a distinct lack of formal work on their properties has provoked many researches to explicitly reject them (Sternefeld 1996; Gärtner 2002; Potts 2002) and led to various conjectures that they are computationally intractable (Collins 1996; Johnson and Lappin 1999). In this paper, I refute the latter 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 lbutts that provide formal models for Focus Economy, Merge-over-Move and the Shortest Derivation Principle. My results shed new light on reference-set computation as well as on OSs and should be of interest to readers from various formal backgrounds, foremost computational phonology and Minimalist Grammars; moreover, when viewed as tree transducers, reference-set constraints also exhibit some previously overlooked connections to synchronous TAG (Shieber and Schabes 1990; Shieber 2004, 2006), the exploration of which I have to leave to future research, unfortunately.

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 defining 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 (Jäger 2002) regarding the properties of an OS which jointly ensure that it does not exceed the power of linear tree transducers. In the last three sections, I show how to model Focus Economy, Merge-over-Move and the Shortest Derivation Principle as such restricted OSs, and I discuss the similarities between the tree transducers corresponding to these constraints.

1 Preliminaries and Notation

Let me introduce some notational conventions first. For any two sets A and B, $A \setminus B$ denotes their relative complement and $A \times B$ their cartesian product. The *diagonal* of A is $id(A) := \{\langle a, a\rangle \mid a \in A\}$. Given a relation $R \subseteq A \times B$, 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 inverse of R is signified by R^{-1}. The *composition* of two relations R and S is $R \circ S := \{\langle a, c\rangle \mid \langle a, b\rangle \in R, \langle b, c\rangle \in S\}$.

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 Gécseg and Steinby (1984) and Kepser and Mönnich (2006) for further details. I also assume that the reader is knowledgeable about string languages and generalized sequential machines (see Hopcroft and Ullman 1979).

Definition 1. A *context-free tree grammar* (CFTG) is 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, $S \in F$ is the start symbol, and 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 $\mathscr{A} := \langle \Sigma, \Omega, Q, Q', \Delta \rangle$, where Σ and Ω are finite ranked alphabets, Q is a finite set of states, $Q' \subseteq Q$ the set of final states, and Δ is a set of productions of the form $f(q_1(x_1), \dots, q_n(x_n)) \rightarrow q(t(x_1, \dots, x_n))$, 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 $\Omega \cup \{x_1, \dots, x_n\}$.

Example. It is easy to write a transducer for very simple cases of wh-movement. First, let the input alphabet Σ consist of all the labels that appear in trees generated by some input grammar. In a GB setting, for instance, Σ contains all lexical items and X′-annotated category labels. The output alphabet contains every element of Σ and the indexed trace t_{wh}. There are only two states, q_* (which we might call the *identity* state) and q_{wh} (which we might call the *I have previously encountered a wh-word*-state). The final state is q_*. The number of productions of the transducer can be rather high $(2 + 3 * |\Sigma - 1|)$), but they can be compressed into 5 production schemes by abstracting away from the specific lexical items. Thus, in the following, σ denotes any element of Σ except the wh-phrase *what*.

$$\sigma \rightarrow q_*(\sigma) \tag{1}$$

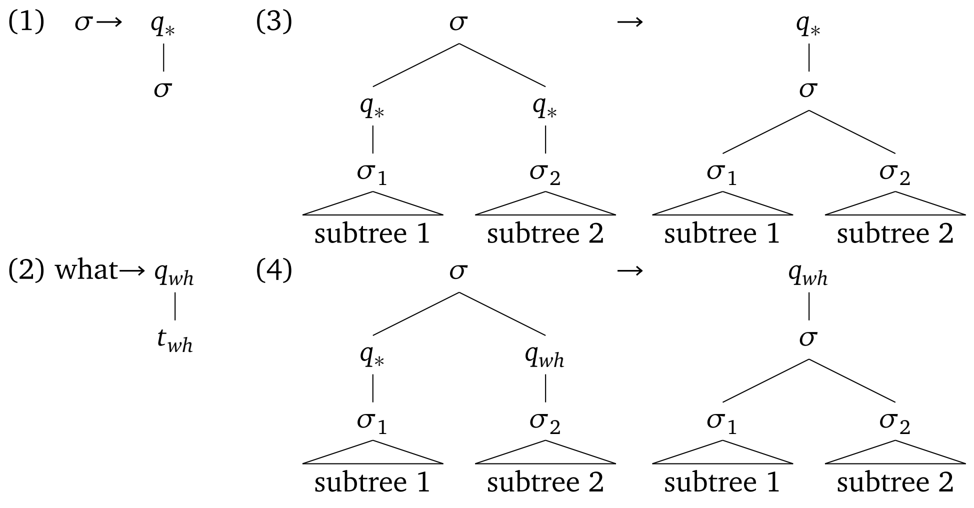
$$\textit{what} \rightarrow q_{wh}(t_{wh}) \tag{2}$$

$$\sigma(q_*(x), q_*(y)) \rightarrow q_*(\sigma(x, y)) \tag{3}$$

$$\sigma(q_*(x), q_{wh}(y)) \rightarrow q_{wh}(\sigma(x, y)) \tag{4}$$

$$\mathrm{TP}(q_i(x), q_{wh}(y)) \rightarrow q_*(\mathrm{CP}(\mathrm{what}, \mathrm{C}'(\mathrm{does}, \mathrm{TP}(x, y)))) \tag{5}$$

These rules can also be written in tree notation.



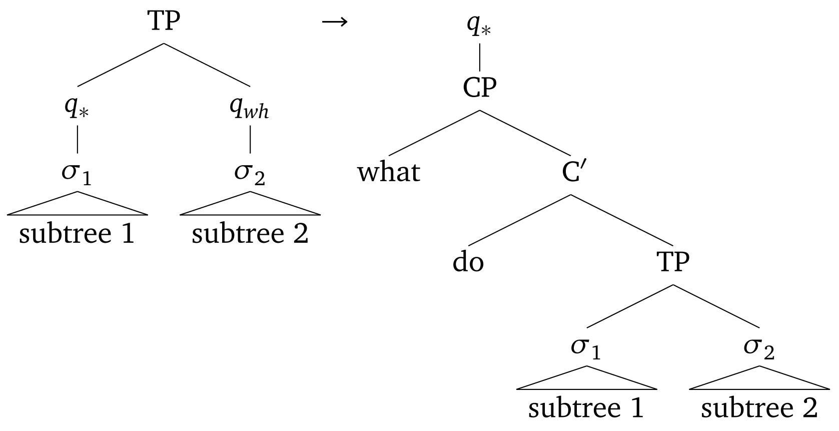
(5)


Figure 1 on the following page shows the phrase structure tree for *the men like what* and how it is transformed by the transducer into the tree for *what do the men like*.

Definition 3. A *top-down tree transducer* is 5-tuple $\mathscr{A} := \langle \Sigma, \Omega, Q, Q', \Delta \rangle$, 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 $q(f(x_1, \ldots, x_n)) \rightarrow t$, where $f \in \Sigma$ is of rank n, $q \in Q$, and t is a tree with the node labels drawn from $\Omega \cup \{q(x) \mid q \in Q, x \in \{x_1, \ldots, x_n\}\}$.

For the sake of succinctness (but to the detriment of readability), I adopt the following notational conventions for tree transducer productions:

- $\alpha_{\{x,y\}}$ is to be read as "α_x or α_y".
- $\alpha_{a \ldots z}(\beta_{a' \ldots z'}, \ldots, \zeta_{a'', \ldots, z''})$ is to be read as "$\alpha_a(\beta_{a'}, \ldots, \zeta_{a''})$ or … or $\alpha_z(\beta_{z'}, \ldots, \zeta_{z''})$".

Example. The production $\sigma(q_{ij\{a,b\}}(x), q_{jkc}(y)) \rightarrow q_{\{a,c\}}(\sigma(x,y))$ is a schema defining eight productions:

$$\begin{array}{ll}
\sigma(q_i(x), q_j(y)) \rightarrow q_a(\sigma(x,y)) & \sigma(q_i(x), q_j(y)) \rightarrow q_c(\sigma(x,y)) \\
\sigma(q_j(x), q_k(y)) \rightarrow q_a(\sigma(x,y)) & \sigma(q_j(x), q_k(y)) \rightarrow q_c(\sigma(x,y)) \\
\sigma(q_a(x), q_c(y)) \rightarrow q_a(\sigma(x,y)) & \sigma(q_a(x), q_c(y)) \rightarrow q_c(\sigma(x,y)) \\
\sigma(q_b(x), q_c(y)) \rightarrow q_a(\sigma(x,y)) & \sigma(q_b(x), q_c(y)) \rightarrow q_c(\sigma(x,y))
\end{array}$$

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 transductions realized by ltdtts is properly contained in the class of transductions realized by 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

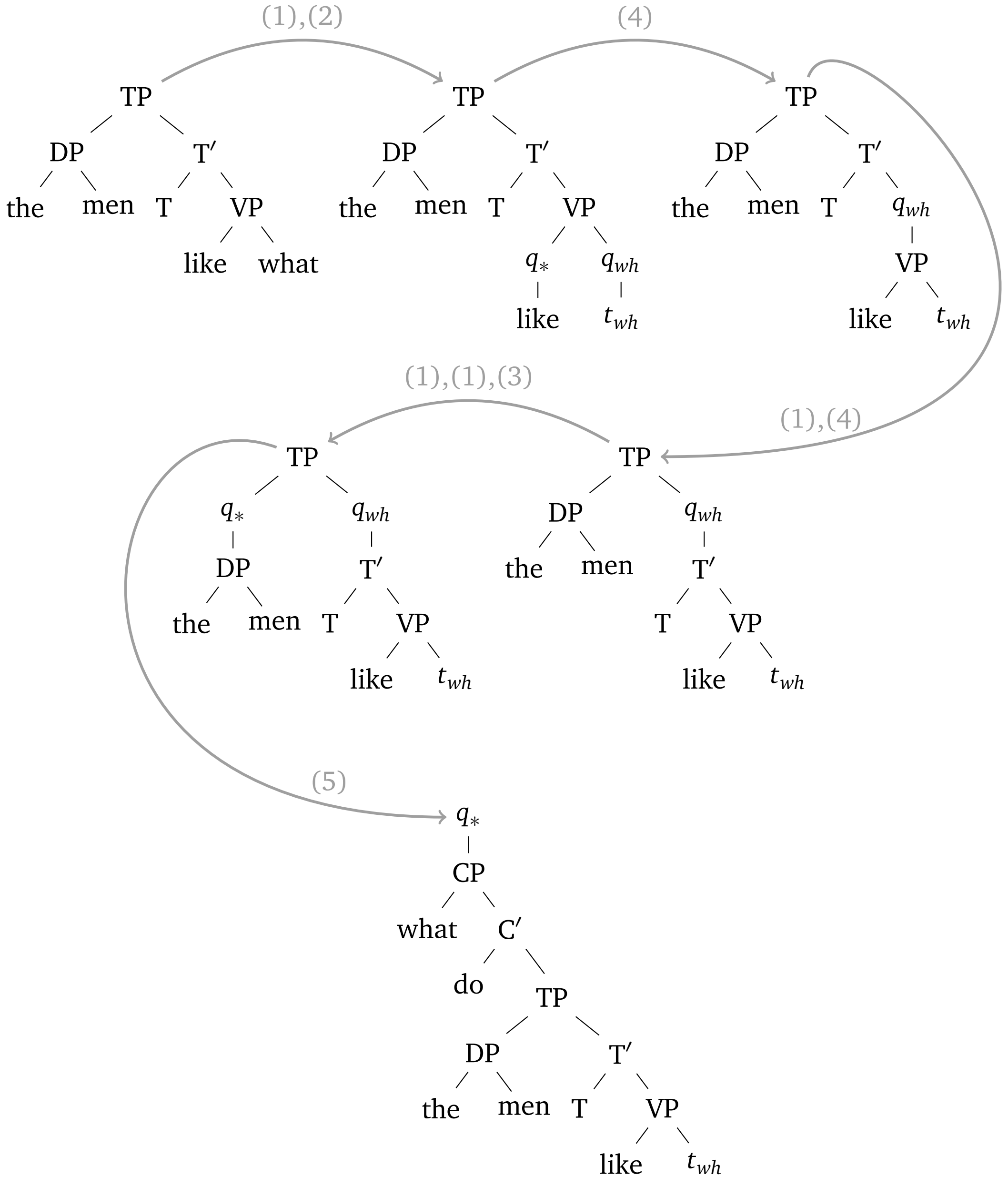


Figure 1: Example of transduction for simple wh-movement

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.

Since reference-set constraints originate from Minimalist syntax, I will often assume that the input language to some transduction is given by a Minimalist grammar (MG).

Definition 4. A *Minimalist grammar* is a 5-tuple $\mathcal{G} := \langle \Sigma, F, Types, Lex, O \rangle$, where

- $\Sigma \neq \emptyset$ is the alphabet,
- F is the union of a non-empty set *base* of *basic features* and its prefixed variants $\{=f \mid f \in base\}$, $\{+f \mid f \in base\}$, $\{-f \mid f \in base\}$ of *selector*, *licensor*, and *licensee* features, respectively,
- $Types := \{::, :\}$ serves in distinguishing *lexical* from *derived* expressions,
- the lexicon *Lex* is a finite subset of $\Sigma^* \times \{::\} \times F^+$,
- and O is the set of generating functions to be defined below.

We define the set C of *chains* $\Sigma^* \times Types \times F^*$ (whence $Lex \subset C$) and refer to non-empty sequences of chains as *expressions*, the set of which we call E.

The set O of generating functions consists of the operations *merge* and *move*. The operation *merge*: $(E \times E) \to E$ is the union of the following three functions, for $s, t \in \Sigma^*$, $\cdot \in Types$, $f \in base$, $\gamma \in F^*$, $\delta \in F^+$, and chains $\alpha_1, \ldots, \alpha_k$, $\iota_1, \ldots, \iota_k$, $0 \leq k, l$:

$$\frac{s :: =f\gamma \qquad t \cdot f, \iota_1, \ldots, \iota_k}{st : \gamma, \iota_1, \ldots, \iota_k}\ \textit{merge1}$$

$$\frac{s : =f\gamma, \alpha_1, \ldots, \alpha_k \qquad t \cdot f, \iota_1, \ldots, \iota_l}{ts : \gamma, \alpha_1, \ldots, \alpha_k, \iota_1, \ldots, \iota_l}\ \textit{merge2}$$

$$\frac{s \cdot =f\gamma, \alpha_1, \ldots, \alpha_k \qquad t \cdot f\delta, \iota_1, \ldots, \iota_l}{s : \gamma, \alpha_1, \ldots, \alpha_k, t : \delta, \iota_1, \ldots, \iota_l}\ \textit{merge3}$$

As the domains of all three functions are disjoint, their union is a function, too.

The operation *move*: $E \to E$ is the union of the two functions below, with the notation as above and the further assumption that all chains satisfy the Shortest Move Constraint (SMC), according to which no two chains in the domain of *move* display the same licensee feature $-f$ as their first feature.

$$\frac{s : +f\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f, \alpha_{i+1}, \ldots, \alpha_k}{ts : \gamma, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \alpha_k}\ \textit{move1}$$

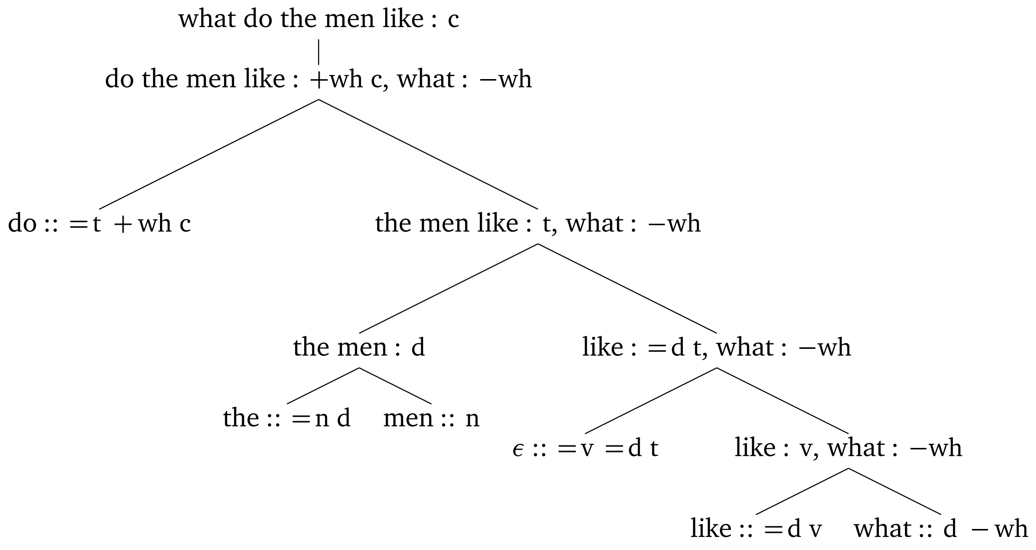
$$\frac{s : +f\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f\delta, \alpha_{i+1}, \ldots, \alpha_k}{s : \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : \delta, \alpha_{i+1}, \ldots, \alpha_k}\ \textit{move2}$$

The language $L(\mathscr{G})$ generated by $\mathscr{G}$ is the closure of the lexicon under the generating functions.

Example. The following MG assigns the question *what do the men like* the same phrase structure tree as the transducer from the previous example.

what :: d −wh	like :: =d v
men :: n	ϵ :: =v =d t
the :: =n d	do :: =t +wh c

The corresponding derivation is depicted below, with binary branching nodes indicating instances of Merge and unary ones instances of Move.



The string language derived by an MG is mildly context-sensitive in the sense of Joshi (1985). In particular, for every MG there exists a strongly equivalent multiple-context free grammar (Michaelis 1998, 2001). This implies that a tree language that can be derived by an MG may not be linear context-free. However, the set of derivation trees of an MG is a regular tree language and there is an effective procedure for obtaining the derived trees from their derivation trees — this holds even of the strictly more powerful class of MGs with unbounded copying (Kobele 2006; Kobele, Retoré, and Salvati 2007).[1]

At the end of Sec. 5.2 and 6.3, I make good use of $\mathscr{L}^2_{K,P}$ (Rogers 1998), 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 $\mathscr{L}^2_{K,P}$ will be briefly introduced in the relevant sections. See Thomas (1997) for further background material on MSO and Rogers (1997, 1998) for $\mathscr{L}^2_{K,P}$ in particular.

[1]An alternative regular representation of MGs is given in Kolb, Michaelis, Mönnich, and Morawietz (2003). The method of Kobele et al. is a better choice for this project, though, as many reference-set constraints in the literature operate on derivation trees.

2 Traditional Perspective on Optimality Systems

OSs were introduced independently by Frank and Satta (1998) and Karttunen (1998) as a formalization of Optimality Theory (OT) (Prince and Smolensky 2004), which has been the dominant theory of phonology in mainstream linguistics for the last fifteen years. 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 (Chomsky and Halle 1968). 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 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. 2, 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, with each c_i preceding c_{i+1}, $1 \leq i < 2$. Constraint c_1 is applied first. Candidates o_2 and o_3 each violate it once, however, o_1 violates it twice and there are no other output candidates. 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 been previously ruled out), 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.

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

Figure 2: Example of an OT evaluation in tableau notation

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 straightforward.

Definition 5. An *optimality system* over languages L, L' is a pair $\mathcal{O} := \langle \textsc{Gen}, C \rangle$ with $\textsc{Gen} \subseteq L \times L'$ and $C := \langle c_1, \dots, c_n \rangle$ a linearly ordered sequence of functions $c_i : \textsc{Gen} \to \mathbb{N}$. For $a, b \in \textsc{Gen}$, $a <_{\mathcal{O}} b$ iff there is an $1 \leq k \leq n$ such that $c_k(a) < c_k(b)$ and for all $j < k$, $c_j(a) = c_j(b)$.

Definition 6. 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 transduction induced by $\mathcal{O}$ is given by $\tau := \{\langle i, o \rangle \mid \langle i, o \rangle$ is optimal with respect to $\mathcal{O}\}$. The output language of $\mathcal{O}$ is $\mathrm{ran}(\tau)$.

The important insight of Frank and Satta (1998) as well as Karttunen (1998), which was later improved upon by Wartena (2000), Jäger (2002) and Kepser and Mönnich (2006), 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 7. *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 regular tree language.*

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

The theorem makes reference to a term the reader is presumably familiar with from the OT literature, output-markedness, which is easily defined in formal terms.

Definition 8. 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 $\langle i, o \rangle, \langle i', o \rangle \in \textsc{Gen}$.

In the case of regular string and tree languages, the theorem can be generalized significantly, as was shown by Jäger (2002).

Theorem 9. *Let $\mathcal{O} := \langle \textsc{Gen}, C \rangle$ be an OS such that*

- $\mathrm{dom}(\textsc{Gen})$ *is regular string/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* $\mathrm{dom}(\tau)$.

Global optimality is a rather technical notion that requires a lot of finite-state machinery to be in place before it can be made formally precise. Intuitively, 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. It is worth going through the formal definition, though, as this will also make it clear why the more general theorem does not carry over to linear context-free tree languages.

We start out with two definitions that reimplement the constraints of an OS, *plus* its filtering procedure in relational terms.

Definition 10. Given some $R \subseteq \textsc{Gen}$, we associate with every $c_i \in C$ a relation $rel_i^R := \{\langle o, o' \rangle \mid c_i(o) < c_i(o')\} \cap (R^{-1} \circ R)$, the *ranking of c_i relativized to R*.

The relativization with respect to R is achieved by intersecting the relation representing the ranking the constraint induces over the entire candidate language with $(R^{-1} \circ R)$, which is the relation that holds between two candidates o and o' iff they are competing output candidates for some input i, i.e. iff both $\langle i, o \rangle$ and $\langle i, o' \rangle$ belong to $\textsc{Gen}$. In structural terms, rel_i^R is the substructure obtained from the structure defined by c_i by removing all branches between output candidates that never compete against each other. Note that since the class of rational string relations is closed under intersection, composition, and taking inverse, rel_i^R will be a rational relation iff R is rational and there is a rational relation S such that $S \cap (R^{-1} \circ R) = \{\langle o, o' \rangle \mid c_i(o) < c_i(o')\} \cap (R^{-1} \circ R)$. Linear tree transductions, on the other hand, are not closed under inverse, so when talking about rational tree relations it has to be ensured that rel_i^R itself is rational.

Definition 11. Let $R \subseteq \textsc{Gen}$ and $c_i \in C$. Then $R \downharpoonright c_i := R \circ id(\mathrm{ran}(R) \setminus \mathrm{ran}(R \circ rel_i^R))$ is called the *generalized lenient composition of R with c_i*.

The generalized lenient composition is the OS-model of the OT-filtering procedure. It looks rather scary but is actually easy to master. Let us proceed from the inside to the outside. By composing R with rel_i^R, we obtain the subset of R in which the output candidates are suboptimal. To see this, suppose that o is an optimal output candidate with respect to rel_i^R, so it is a minimal element in the structure defined by rel_i^R. Now suppose that there is an output candidate o' that is worse than o, i.e. $\langle o, o' \rangle \in rel_i^R$. Assuming that R contains the pair $\langle i, o \rangle$, composing R and rel_i^R means composing the pairs $\langle i, o \rangle$ and $\langle o, o' \rangle$, which yields $\langle i, o' \rangle$. The optimal candidate o has been "hidden" by the composition. With this short explanation the reader should now be able to verify that the range of $R \circ rel_i^R$ contains all suboptimal output candidates. Removing those from the range of R and taking the diagonal over this new set thus yields the identity function over the set of optimal output candidates, and composing R with this function hence means filtering out all suboptimal candidates (as a helpful exercise, the reader may verify that we get the right result even if all competing output candidates incur the same number of violations with respect to c_i).

To see that the generalized lenient composition is a finite-state procedure, it suffices to know (i) that the range of a rational relation is a regular language if its domain is regular, (ii) that the class of regular languages is closed under relative complement, and (iii) that the diagonal of a regular set is a rational relation. However, the class of linear context-free languages is not closed under relative complement, and this is why we need stronger conditions for OSs over linear context-free languages.

The finite-state perspective on OSs also allows for a redefinition of the transductions they induce. As Jäger (2002) shows, transductions can be defined purely in terms of generalized lenient composition.

Theorem 12. *Given an OS $\mathcal{O} := \langle \textsc{Gen}, \langle c_1, \ldots, c_n \rangle \rangle$ satisfying the conditions above, $\tau := \textsc{Gen} \upharpoonright c_1 \upharpoonright \ldots \upharpoonright c_n$.*

But we set out to give a formal definition of global optimality, and now we have all the required machinery.

Definition 13. Let R and S be relations. *Optimality is global with respect to R and S* iff

$$\forall i, o[(iRo \wedge \neg\exists p(iRp \wedge pSo)) \rightarrow \neg\exists p(p \in \operatorname{ran}(R) \wedge pSo)]$$

Now let $\mathcal{O} := \langle \textsc{Gen}, C \rangle$, $C := \langle c_1, \ldots, c_n \rangle$ be an optimality system. Then we say that $\mathcal{O}$ is *globally optimal* or *satisfies global optimality* iff optimality is global with respect to $\textsc{Gen} \upharpoonright c_1^{i-1} := \textsc{Gen} \upharpoonright c_1 \upharpoonright \ldots \upharpoonright c_{i-1}$ and the ranking of c_i relativized to $\textsc{Gen} \upharpoonright c_1^{i-1}$ for all $1 \leq i \leq n$.

The important thing to keep in mind here is that S does not correspond to the absolute ranking induced by a constraint c_i, but its relativized ranking (i.e. the one with “holes” in it). Therefore global optimality does not demand that if o is optimal for i, there is no output p that is more optimal than o for i, but rather that no output p that competes against o with respect to some input j is more optimal than o. In less technical terms, if o is an optimal output candidate for some input i, then there is no input j for which o is an output candidate but not an optimal one.

In the next section, I introduce a notational variant of OS and use this variant to show that global optimality is always satisfied for reference-set constraints, thereby highlighting them as a very restricted subclass of OS. For the sake of simplicity, I will implicitly assume that the OSs consist of only one constraint, as this does away with a lot of tedious induction steps. Fortunately, this has no negative consequences for the relevance of the results. First, when modelling reference-set constraints we do not need more than one constraint. Second, every OS $\mathcal{O} := \langle \textsc{Gen}, \langle c_1, \ldots, c_n \rangle \rangle$ can be decomposed into a cascade of n OSs $\mathcal{O}_k := \langle \textsc{Gen}_k, \langle c_k \rangle \rangle$ such that $\textsc{Gen}_k$ is given by $\textsc{Gen}$ for $k = 1$ and $\{\langle i, o \rangle \in \textsc{Gen}_{k-1} \mid \langle i, o \rangle$ is optimal with respect to $c_{k-1}\}$ otherwise (for finite-state OS, the latter is equivalent to $\textsc{Gen}_{k-1} \upharpoonright c_{k-1}$).

3 Controlled Optimality Systems

OSs are perfectly capable of modelling reference-set constraints. The reference set for any input i is defined by $i\textsc{Gen}$, 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 $\textsc{Gen}$ 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. Focus Economy is slightly more involved and allows for different models. In one of them (which isn't the one I will formalize in Sec. 5.2), Gen relates a tree t without stress annotation but a focus-marked constituent to its analogs with neutral and shifted stress. The OS then uses two constraints, the first of which weeds out trees in which the focus-marked constituent does not contain the main stress carrier, while the second penalizes derivation length (the assumption here is that derivations for trees with shifted stress are longer than those for trees with neutral stress). Then a tree derived by stress shift will be optimal just in case the equivalent tree with neutral stress has already been ruled out by the first constraint.

While these two examples show that OSs certainly 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. Note, though, that in the case of a constraint like Focus Economy, where input language and candidate language are disjoint, distinct inputs might be assigned the same reference set. In such a configuration, 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 14. An *$\mathscr{F}$-controlled optimality system* over languages L, L' is a 4-tuple $\mathcal{O}[\mathscr{F}] := \langle \textsc{Gen}, C, \mathscr{F}, \gamma \rangle$, where

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

Every controlled OS can be translated into a canonical OS by discarding its third and fourth component (i.e. $\mathscr{F}$ and the control map γ). In the other direction, a controlled OS can be obtained from every canonical OS by setting $\mathscr{F} := \{\{i\} \mid i \in L\}$ and $\gamma : \{i\} \mapsto i\textsc{Gen}$. 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 — recall that 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 may 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. 3 on the facing 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 (see Fig. 4 on the next page). 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.

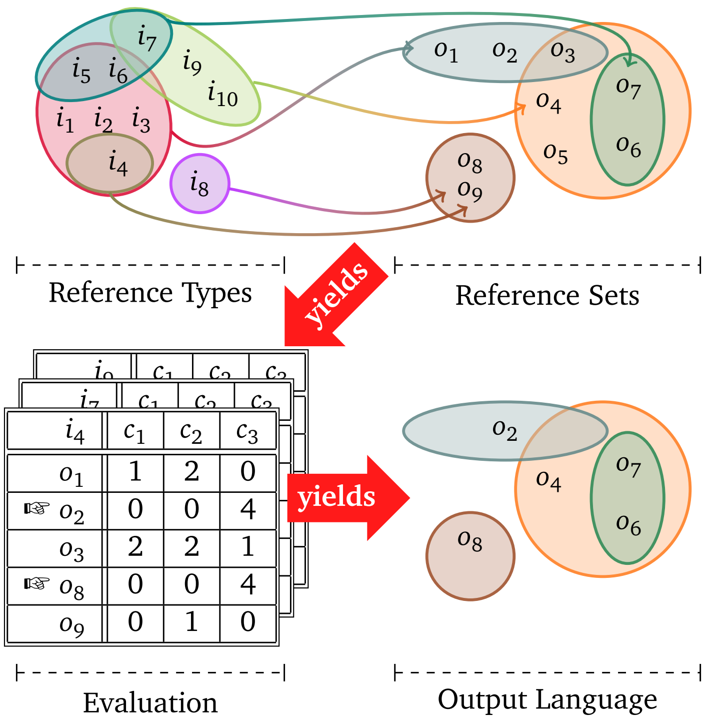
The Fewest Steps-type class of OSs is best captured by the notion of *endocentricity*.

Definition 15. An $\mathscr{F}$-controlled OS is *endocentric* iff $X\gamma = X$ for all $X \in \mathscr{F}$.

Constraints like Focus Economy, on the other hand, are *output-partitioned* and *output-segregated* in the following senses:

Definition 16. An $\mathscr{F}$-controlled OS is

- *output-partitioned* iff for all distinct $X, Y \in \mathscr{F}$, $X\gamma \neq Y\gamma$ implies $X\gamma \cap Y\gamma = \emptyset$.
- *output-segregated* iff for all distinct $X, Y \in \mathscr{F}$, neither $X\gamma \subseteq Y\gamma$ nor $Y\gamma \subseteq X\gamma$.



i_4	c_1	c_2	c_3
o_1	1	2	0
☞ o_2	0	0	4
o_3	2	2	1
☞ o_8	0	0	4
o_9	0	1	0

Figure 3: 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

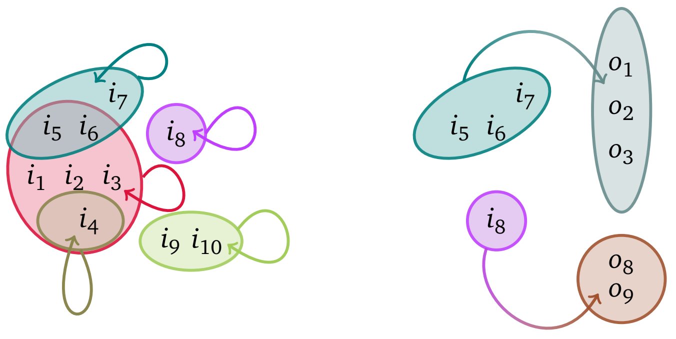


Figure 4: Almost all instances of reference-set computation in the literature use one of the two configurations above, both of which are output joint preserving

While this class certainly fits Focus Economy and its ilk pretty well, it does not extend naturally to endocentric OSs, which means that we would have to study the two classes independently from each other. In particular, endocentric OSs aren't necessarily output-partitioned, as they do allow for overlapping reference-sets. Nor is output-segregation a meaningful restriction on endocentric OSs: Given some $\mathscr{F}$-controlled OS, assume $x \in X$, $y \in Y$, $X \subseteq Y$. Then x is also a member of Y, whence both $\langle x, y \rangle$ and $\langle y, x \rangle$ are in (the graph of) Gen. The reference type X is immaterial for computing Gen and can be removed from $\mathscr{F}$ without consequences. Given these discrepancies, we should not be too sure that results pertaining to one class can easily be carried over to the other. As it turns out, though, a natural unification of the two subclasses is available in the form of *output joint preservation*.

Definition 17. An $\mathscr{F}$-controlled optimality system is *output joint preserving* iff for all distinct $X, Y \in \mathscr{F}$, $X\gamma \cap Y\gamma \neq \emptyset \rightarrow X \cap Y \neq \emptyset$.

The OS depicted in Fig. 3 on the preceding page fails output joint preservation. It is clearly violated by Sienna and Purple, which are disjoint yet mapped to the same reference set, Brown. It isn't respected by Red and Lime, either, which are mapped to Blue and Orange, respectively, the intersection of which is non-empty even though Red and Lime are disjoint. Moving on to Fig. 4 on the previous page, we observe that every endocentric OS is output joint preserving. In addition, all output-partitioned OSs trivially satisfy output joint preservation, too, because there are no joints to preserve in these OSs. These inclusions tell us that 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 in conjunction with another property it is also sufficiently restrictive to establish a link to global optimality.

Definition 18. An $\mathscr{F}$-controlled OS is *type-level optimal* iff $X\tau^{\uparrow} \restriction X\gamma = X\tau^{\downarrow} \restriction X\gamma$ for all $X \in \mathscr{F}$.

Type-level optimality is essentially the restriction of global optimality to reference types: If o is an optimal output candidate for some $x \in X$, then there is no $y \in X$ such that o isn't optimal for y. Against this backdrop, the following lemma is hardly surprising.

Lemma 19. *Let $\mathscr{O}[\mathscr{F}]$ an $\mathscr{F}$-controlled OS. Then $\mathscr{O}[\mathscr{F}]$ is type-level optimal if it is globally optimal.*

Proof. We prove the contrapositive. If $\mathscr{O}[\mathscr{F}]$ is not type-level optimal, then it holds for some $X \in \mathscr{F}$ 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. □

It should also be pointed out that type-level optimality is trivially satisfied if all reference types are singleton. This setting also provides numerous examples that show that the converse of the lemma does not hold. For instance, let $\mathscr{O}[\mathscr{F}]$ consist only of the reference types $\{i\}$ and $\{j\}$, which are both mapped to the reference set $\{o, p\}$, but $i\tau = \{o\}$ whereas $j\tau = \{p\}$. Then $\mathscr{O}[\mathscr{F}]$ is type-level optimal but not globally optimal. The problem is that type-level optimality leaves a loop-hole for OSs, as different reference types may have overlapping reference sets but disagree on which candidates in the intersection they deem optimal. This hole is patched by output joint preservation.

Theorem 20. *Every output joint preserving OS is type-level optimal iff it is globally optimal.*

Proof. The right-to-left direction follows from Lem. 19. 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 $\langle x,z \rangle, \langle y,z \rangle \in \textsc{Gen}$ 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 assume towards a contradiction that $\mathcal{O}[\mathcal{F}]$ 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. □

To reiterate, 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 candidate, there exists some input p belonging to both X and Y that will be faced with 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 is clear, then, 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 interfere with type-level optimality. This is indeed feasible thanks to the following entailment.

Lemma 21. *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. □

Corollary 22. *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. 7 for $\mathcal{F}$-controlled OSs.

Corollary 23. *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)/*a rational tree language, and*
- *$\mathcal{O}[\mathscr{F}]$ 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 modelling reference-set constraints, they also have the neat property that the global optimality check is redundant, thanks to Lem. 21. As it is pretty easy to determine for any given reference-set constraint whether it can be modeled by output-markedness constraints alone, the only stumbling block in the implementation of constraints that can be modeled by output joint preserving OSs is the transducers for the constraints and Gen. If those transducers each define a rational relation, so does the entire optimality system.

It is not difficult to see, however, that Thm. 20 leaves ample room for generalization. After all, the only role of output joint preservation is to ensure — in a rather round-about way — that reference types with overlapping reference sets agree on which candidates in the intersection they consider optimal. If this criterion can be expressed directly, output joint preservation is redundant.

Definition 24. An $\mathscr{F}$-controlled OS $\mathcal{O}[\mathscr{F}]$ satisfies *synchronized optimality* or is *in sync* iff it is type-level optimal and satisfies the following condition:

(*) for all $X, Y \in \mathscr{F}$ with $X\gamma \cap Y\gamma \neq \emptyset$, if $z \in X\gamma \cap Y\gamma$ is an optimal output candidate for X, it is also optimal for Y.

Theorem 25. *An $\mathscr{F}$-controlled OS $\mathcal{O}[\mathscr{F}]$ satisfies synchronized optimality if and only if it is globally optimal.*

Proof. In proving the contrapositive of the left-to-right direction, we distinguish two cases. If it holds for all $X, Y \in \mathscr{F}$ that $X\gamma \cap Y\gamma = \emptyset$, then (*) is vacuously satisfied but type-level optimality does not hold since $\mathcal{O}[\mathscr{F}]$ fails global optimality, so there has to be a reference type $X \in \mathscr{F}$ whose inputs disagree on the optimality of some $o \in X\gamma$. So assume that $\mathcal{O}[\mathscr{F}]$ is type-level optimal but not output-partitioned. Then the lack of global optimality entails that types $X, Y \in \mathscr{F}$ disagree on the optimality of some $o \in X\gamma \cap Y\gamma$, whence (*) is not satisfied.

It only remains for us to show the implication in the other direction. We prove the contrapositive. If $\mathcal{O}[\mathscr{F}]$ is not in sync, then it fails type-level optimality or violates (*). In the former case, Lem. 19 tells us immediately that the OS does not satisfy global optimality. Assume then w.l.o.g. that $\mathcal{O}[\mathscr{F}]$ is type-level optimal but (*) does not hold. Then there are $X, Y \in \mathscr{F}$ and $z \in X\gamma \cap Y\gamma$ such that $x\text{Gen} \ni z \in y\text{Gen}$ for some $x \in X$ and some $y \in Y$, yet $x\tau \ni z \notin y\tau$. Thus $\mathcal{O}[\mathscr{F}]$ is not globally optimal. □

While Thm. 25 is more general than Thm. 20, its use is also more limited for practical purposes. This is because synchronized optimality does not follow from restricting the OS to output-markedness constraints. Just consider a case where $X\gamma$ is a proper subset of $Y\gamma$. Then it cannot be precluded that even though $x \in X\gamma$ is optimal for X, there is some $y \in Y\gamma \setminus X\gamma$ that it loses out against, which means that x is not optimal for Y. In particular, if $X\gamma$ is singleton, x will always be optimal for X, no matter how bad a candidate it is. Without the entailment from output-markedness constraints, we are forced to manually check for synchronized optimality, which might be a laborious process. As output joint preservation seems to be a robust property of reference-set constraints and makes it a lot easier to check for global optimality, Thm. 20 is of greater importance.

4 Transduction Preserving Operations

One argument that could be leveled against approaching global optimality by means of the subclass of output joint preserving OSs rather than through synchronized optimality is that output-joint preservation does not extend to several configurations that seem very natural and may in principle give rise to globally optimal OSs. Consider for instance an $\mathscr{F}$-controlled OS where $\mathscr{F}$ consists only of two disjoint reference types X and Y, which are mapped to the same reference set. Such an OS fails output joint preservation, yet it might be globally optimal if it is type-level optimal.

One reply to this concern is to emphasize once more that such configurations do not seem to occur with reference-set constraints. However, such a rebuttal would be rather unappealing on a theoretical level, especially because the solution to the problem above is simple: if we take the union of the reference types X and Y, what we get is an OS that defines the same generator relation, and thus the same transduction, yet satisfies output joint preservation. So it seems that output joint preservation is not as restrictive a property as one might think, provided that we allow ourselves to meddle with the makeup of $\mathscr{F}$. The question we have to answer, then, is in which ways $\mathscr{F}$ may be manipulated without affecting the transduction induced by an OS.

Note that the malleability of $\mathscr{F}$ is interesting for practical purposes, too, as it might allow us to deal with peculiar cases where some reference types taken by themselves are considerably more complex than their union. To give an example, let P be the set of strings over some alphabet Σ whose length is prime, and Q its complement. Now P does not define a regular language, not even a context-free one (it is context-sensitive), but the union of P and Q is Σ^*, which is regular (in fact, it is strictly 2-local, i.e. it belongs to one of the weakest classes of string languages that are commonly studied).

So let us see what kind of operations can be applied to $\mathscr{F}$ without altering the transduction τ induced by the OS. In our short discussion above, it was already mentioned that if an operation has no effect on GEN, it has no effect on τ either. In the case at hand, the operation was to take unions of reference types that are mapped to the same reference set. Or speaking in functional terms, we took the union of all reference types that have the same image under the control map γ. So we only turned γ from a many-to-one into a one-to-one function.

Theorem 26. *Let $\mathcal{O}[\mathscr{F}] := \langle \textsc{Gen}, C, \mathscr{F}, \gamma \rangle$ be an $\mathscr{F}$-controlled OS over language L, L′. Then there is an $\mathscr{F}'$-controlled OS $\mathcal{O}[\mathscr{F}'] := \langle \textsc{Gen}, C, \mathscr{F}', \gamma' \rangle$ over the same languages such that γ' is one-to-one (and $\mathscr{F}'$ is obtained from $\mathscr{F}$ by taking the union of all $X, Y \in \mathscr{F}$ with $X\gamma = Y\gamma$).*

Thanks to this theorem, only OSs whose control map is one-to-one need to be considered in the remainder of this section. In these cases taking unions of reference types also requires taking union of reference sets. So if $X\gamma = A$ and $Y\gamma = B$, we want $(X \cup Y)\gamma$ to be $A \cup B$. The same holds in the other direction, whence it does not make a real difference whether we talk about unions of reference types or reference sets. Soon though it will become evident that the conditions one may want to impose are best expressed over reference sets; consequently, "taking unions" should be read as "taking unions of reference sets" in the following.

Many examples are readily at hand that establish that transductions are not preserved under arbitrary union. Consider an OS with $\mathscr{F} := \{\{a, b\}, \{b, c\}\}$, $\{a, b\}\gamma = \{d\}$, $\{b, c\}\gamma = \{e\}$, $a\tau = \{d\}$, $c\tau = \{e\}$, and $b\tau = \{d, e\}$. If we take the union of $\{d\}$ and $\{e\}$ (and consequently also of $\{a, b\}$ and $\{b, c\}$), there is no guarantee that the transduction will remain the same. In fact, it is very unlikely, since the constraints of the OS would have to be sensitive to the input in such a way that d and e are equally optimal for b, but d is a better output for a and at the same time worse for c. This is a feasible out-turn, but hardly a common one. In particular, there is no way such a result could obtain if all constraints were output-markedness constraints, which is an important restriction for us. It also implies that we lost global optimality by unionizing the reference sets. In the light of this outcome it seems prudent to focus our attention on globally optimal OSs that use only output-markedness constraints; after all, we are mostly interested in tinkering with computable OSs and extending the applicability of output joint preservation, so these restrictions have no negative repercussions for our endeavor.

For globally optimal OSs with output-markedness constraints only, then, arbitrary union will in general induce a change in the transduction. As indicated by the example above, this is almost inevitable if the reference sets are disjoint, since it is very unlikely that all optimal output candidates of reference set A incur the same number of violations with every constraint as all optimal output candidates of reference set B. So what about taking unions of overlapping reference sets? Can we find restrictions for this case that will ensure that the transduction itself remains untouched?

At first sight overlapping reference sets appear to be just as problematic. If we add an element f to the reference sets in the previous example, we run into the same problems nonetheless. The source of all our troubles isn't the disjointness of the reference sets, but that by unionizing the reference sets, the optimal output candidates of reference set A are suddenly confronted with new contenders, the optimal output candidates of reference set B; unless all optimal output candidates of A and B are evenly matched, a change in the transduction is inevitable. Figuratively speaking, taking unions is a little bit like reshuffling the pack. For disjoint reference sets, the reshuffling always creates an unpredictable situation, but with overlapping reference sets, there is a case where we can still predict the outcome after the reshuffling: if all optimal output candidates already had to compete against each other in the original OS.

Definition 27. Given an OS $\mathcal{O}[\mathscr{F}] := \langle \textsc{Gen}, \langle c_1, \ldots, c_n \rangle, \mathscr{F}, \gamma \rangle$ over L, L', $z \in L'$ is a *finalist* iff it is in the range of $\textsc{Gen} \downharpoonleft c_1 \downharpoonleft \ldots \downharpoonleft c_{n-1}$. Two sets $X, Y \in \mathscr{F}$ with $X\gamma \cap Y\gamma \neq \emptyset$ are *finalist intersective* iff $X\gamma \cap Y\gamma \supseteq \{z \in X\gamma \cup Y\gamma \mid z \text{ is a finalist}\}$. An optimality system is *finalist intersective* iff all $X, Y \in \mathscr{F}$ with $X\gamma \cap Y\gamma \neq \emptyset$ are finalist intersective.

The notion of being finalist intersective is rather indirect, ideally there would be a particular arrangement of reference sets that can be linked to it in a systematic way. Now observe that no optimality system $\mathcal{O}[\mathscr{F}]$ with distinct $X, Y, Z \in \mathscr{F}$ such that $X\gamma \cap Y\gamma \neq \emptyset$, $X\gamma \cap Z\gamma \neq \emptyset$ and $X\gamma \cap Y\gamma \cap Z\gamma = \emptyset$ is finalist intersective. Otherwise, all finalists of X would have to be contained in $X\gamma \cap Y\gamma$ and $X\gamma \cap Z\gamma$, i.e. $(X\gamma \cap Y\gamma) \cap (X\gamma \cap Z\gamma) = X\gamma \cap Y\gamma \cap Z\gamma$, which is the empty set. In fact it can be shown that all finalist intersective OSs are built from one basic pattern, which I call a blossom.

Definition 28. Given an $\mathscr{F}$-controlled OS $\mathcal{O}[\mathscr{F}]$ with $n \geq 1$ distinct $X_1, \ldots, X_n \in \mathscr{F}$ such that $\bigcap_{i=1}^{n} X_i\gamma$ is non-empty, we call $B := \{X_1\gamma, \ldots, X_n\gamma\}$ a *blossom* of $\mathcal{O}[\mathscr{F}]$, each $X_i\gamma$ a *petal* of B and $\bigcap_{i=1}^{n} X_i\gamma$ the *stem* of B. We say that B is maximal iff there is no distinct $X_{n+1} \in \mathscr{F}$ with $X_{n+1}\gamma \cap X_i\gamma \neq \emptyset$.

From the definition it follows immediately that all distinct maximal blossoms are disjoint.

Lemma 29. *Let $\mathcal{O}[\mathscr{F}]$ be an $\mathscr{F}$-controlled OS with $B := \{X_1, \ldots, X_n\} \subseteq \mathscr{F}$, $n \geq 1$. If $X_1\gamma, \ldots, X_n\gamma$ are pairwise finalist intersective then B is a blossom, the stem of which contains the finalists of each X_i, $1 \leq i \leq n$.*

Proof. If $\mathcal{O}[\mathscr{F}]$ is output partitioned, this is trivially true. In all other cases, we have to show that $\bigcap_{i=1}^{n} X_i$ is non-empty and that it contains every finalist. This follows from a simple proof by induction: W.l.o.g. put all members of B in a total order that is reflected by their index. Now consider X_1 and X_2 in B. Since they are finalist intersective, $X_1 \cap X_2 \neq \emptyset$ and contains all finalists of X_1 and X_2. Now suppose that M is the intersection of $X_1, \ldots, X_m \in B$, $2 < m < n$; by assumption it contains the finalists of each X_i. By virtue of pairwise finalist intersectivity, $X_{m+1} \cap X_i$ has to contain all the finalists of X_{m+1} as well as X_i for every $1 \leq i \leq m$, so $X_{m+1} \cap M$ is non-empty and contains all the finalists of $X_1, \ldots, X_{m+1}$. □

Theorem 30. *Let $\mathcal{O}[\mathscr{F}]$ be an OS that satisfies global optimality and uses only output-markedness constraints. Then global optimality of $\mathcal{O}[\mathscr{F}]$ is preserved under union of finalist intersective reference sets.*

Proof. Assume w.l.o.g. that the reference sets $X_1, \ldots, X_n$ are finalist intersective. By the previous lemma, all finalists of each X_i, $1 \leq i \leq n$, belong to $\bigcap_{i=1}^{n} X_i$. So none of them are outranked by any member of $X_j \setminus \bigcap_{i=1}^{n} X_i$ for all $1 \leq j \leq n$ (an easy proof by contradiction is sufficient to establish this). Thus no member of $\bigcup_{i=1}^{n} X_i \setminus \bigcap_{i=1}^{n} X_i$ can be an optimal output candidate for any $x \in \bigcup_{i=1}^{n} X_i$, whence $x\tau$ is unaffected by extending $x\textsc{Gen}$ to $\bigcup_{i=1}^{n} X_i$ and global optimality is preserved. □

Since maximal blossoms are disjoint, it follows that if an OS is finalist intersective, an equivalent output-partitioned OS can be obtained by taking unions of finalist intersective reference sets. Adding to this the observation that γ can be assumed to be one-to-one, we derive that every finalist intersective OS has an equivalent output joint preserving OS.

Corollary 31. *For every finalist intersective OS* $\mathcal{O}[\mathscr{F}] := \langle \textsc{Gen}, C, \mathscr{F}, \gamma \rangle$ *there exists an output joint preserving OS* $\mathcal{O}[\mathscr{F}]' := \langle \textsc{Gen}, C, \mathscr{F}', \gamma' \rangle$ *that defines the same transduction.*

5 Focus Economy

Judging from the general results about OSs it seems very likely that many reference-set constraints can be implemented by linear transducers and hence are efficiently computable. We already established that output joint preservation is usually satisfied, and the same goes for type-level optimality. The crucial question, then, is whether Gen and the constraints can be computed by linear transducers. In the remainder of this paper, I demonstrate that this is true for three popular constraints, starting with Focus Economy.

5.1 *Focus Economy Explained*

Focus Economy (Szendrői 2001; Reinhart 2006) 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 Paul bought a new **car**.
b. My friend Paul **bought** a new car.
c. My friend Paul bought a **new** car.

That these utterances are associated to different information structures is witnessed by the following (in)felicity judgments. For the reader's convenience, the focus, i.e. the new discourse material introduced by each answer, is put in square brackets.

(3) What happened?
a. [$_F$My friend Paul bought a red **car**.]
b. # [$_F$My friend Paul **bought** a red car.]
c. # [$_f$My friend Paul bought a **red** car.]

(4) What did your friend Paul do?
a. He [$_F$ bought a red **car**].
b. # He [$_F$ **bought** a red car].
c. # He [$_F$ bought a **red** car].

(5) What did your friend Paul buy?
a. He bought [$_F$ a red **car**].

b. # He **bought** [$_F$ a red car].

c. # He bought [$_F$ a **red** car].

(6) Did your friend Paul sell a red car?

a. # No, he [$_F$ bought] a red **car**.

b. No, he [$_F$ **bought**] a red car.

c. # No, he [$_F$ bought] a **red** car.

(7) Did your friend Paul buy a green car?

a. # No, he bought a [$_F$ red] **car**.

b. # No, he **bought** a [$_F$ red] car.

c. He bought a [$_F$ **red**] car.

Restricting our attention to the a-sentences only, we might conclude that a constituent can be focused just in case one of its subconstituents carries sentential main stress. A short glimpse at the b- and c-utterances falsifies this conjecture, though. Perhaps, then, main stress has to fall on the subconstituent at the right edge of the focused constituent? This is easily shown to be wrong, too. In (8) below, the stressed constituent isn't located at either edge of the focused constituent.

(8) a. What happened to Mary?

b. [$_F$ John **killed** her.]

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 (Reinhart 2006: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. 5 for an example.

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" (Reinhart 2006: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** Paul bought a new red car" is obtained from the tree in Fig. 5 by changing friend$_W$ and Paul$_S$ to friend$_S$ and Paul$_W$, 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 (8), this burden is put on a separate rule, for independent

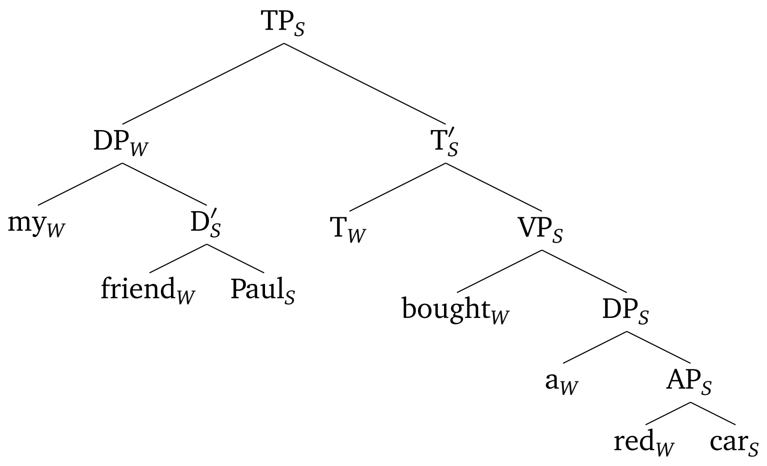


Figure 5: The stress-annotated phrase structure tree for (2a)

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" (Reinhart 2006:p.147). Thus Anaphoric Destressing not only accounts for the unstressed anaphor in (8), but also for the special behavior of stress in cases of parallelism.

(9) 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.

(10) *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 "Paul bought a red **car**", for instance, contains [car], [.AP red car], [.DP a red car], [.VP bought a red car] and [.TP Paul bought a red car] (equivalently, we could associate every node in the tree with a unique address and simply use these addresses in the focus set). 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.

(11) *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.

5.2 A Model of Focus Economy

After this general overview, we may attempt 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 Reinhart (2006), so there is no original material to formalize. Incidentally, movement seems to introduce several interesting complications, as illustrated by sentences involving topicalization, where no other assignment of focus and main stress is grammatical.

(12) a. [$_F$ **John**$_i$] Paul likes t$_i$.
b. * John$_i$ [$_F$**Paul**] likes t$_i$.
c. * John$_i$ [$_F$ Paul **likes** t$_i$].

At the end of the section I argue that the model can be extended to capture theories involving movement.

The last simplification concerns Anaphoric Destressing itself. While the core of Anaphoric Destressing, the destressing of pronominal (and possibly covert) elements, is easy to accommodate, the more general 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 destressed 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 MG $\mathcal{E}$ for English (Stabler and Keenan 2003) in which interior nodes are given explicit category labels (once more for the sake of convenience). As MGs without remnant movement generate regular tree languages (Kobele 2010), it is safe to assume that MGs without any kind of movement do so, too.

Next I define Gen as the composition of four linear transducers corresponding to Anaphoric Destressing, the Main Stress Rule, Stress Shift, and Focus Projection, respectively. Given a tree t derived by $\mathscr{E}$, 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 later step.

Anaphoric Destressing 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 32. Let $\Sigma := \Sigma_L \cup \Sigma_A$ be the vocabulary of the MG $\mathscr{E}$ 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 $\mathscr{D}$ where $\Sigma_{\mathscr{D}} := \Sigma$, $\Omega_{\mathscr{D}}$ is the union of Σ and $\Sigma_D := \{\sigma_D \mid \sigma \in \Sigma\}$, $Q := \{q_i, q_d\}$, $Q' := \{q_i\}$, and $\Delta_{\mathscr{D}}$ contains the rules below, with $\sigma \in \Sigma$ and $\sigma_D \in \Sigma_D$:

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

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 33. *Main Stress* is the lbutt $\mathscr{M}$ where $\Sigma_{\mathscr{M}} := \Omega_{\mathscr{D}}$, $\Omega_{\mathscr{M}}$ 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 $\Delta_{\mathscr{M}}$ contains the following rules, with $\sigma \in \Sigma$, $\sigma_A \in \Sigma_A$, $\sigma_x \in \{\sigma_x \mid \sigma \in \Sigma\}$ for $x \in \{D, S, W\}$:

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

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

Definition 34. *Stress Shift* is the ltdtt $\mathscr{S}$ where $\Sigma_{\mathscr{S}} = \Omega_{\mathscr{S}} = \Omega_{\mathscr{M}}$, $Q := \{q_i, q_s, q_w\}$,

$Q' := \{q_s\}$, and $\Delta_{\mathscr{S}}$ contains the rules below, with $\sigma \in \Sigma_{\mathscr{S}}$ and $\sigma_* \in \Sigma_*$:

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

The last component is Focus Projection, which is formalized as 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 35. *Focus Projection* is the ltdtt $\mathscr{F}$, where $\Sigma_{\mathscr{F}} = \Omega_{\mathscr{S}}$, $\Omega_{\mathscr{F}}$ is the union of $\Omega_{\mathscr{S}}$ and $\Omega^F_{\mathscr{S}} := \{\omega^F \mid \omega \in \Omega_{\mathscr{S}}\}$, $Q := \{q_f, q_g\}$, $Q' := \{q_f\}$, and $\Delta_{\mathscr{F}}$ contains the rules below, with $\sigma \in \Sigma_{\mathscr{F}}$ and $\sigma_{\overline{S}} \in \Sigma_{\mathscr{F}} \setminus \{\sigma_S \mid \sigma \in \Sigma\}$:

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

All four transducers are linear, so they can be composed into a single lbutt modelling Gen (see Engelfriet 1975 for a constructive proof that every ltdtt can be converted into an lbutt defining the same transduction). 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 $\mathscr{L}^2_{K,P}$ (Rogers 1998). 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, $x \triangleleft y$ denotes "x is the parent of y", $x \prec y$ "x is the left sibling of y", and $\triangleleft^*$ the reflexive transitive closure of $\triangleleft$.

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

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

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

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 aren't contained in the FocusPath. This is formally expressed by the $\mathscr{L}^2_{K,P}$ 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 $\omega^F \in \Omega^F_{\mathscr{S}}$. I furthermore use $X \approx Y$ as a shorthand for $\forall x[X(x) \longleftrightarrow Y(x)]$.

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

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. That is, Focus Economy preserves the regularity of the input 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 an MG without movement. But note that MGs 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 (Kobele 2010). Now keep in mind that regular tree languages yield context-free string languages, which are generally assumed to be powerful enough for the greatest part of natural language syntax. Thus the restriction to regular tree languages itself does not preclude us from accommodating most instances of movement.

If we want the full expressive power of MGs, then the best strategy is to express Focus Economy as a constraint over derivation trees, since for every MG the set of derivation trees it licenses forms a regular language that fully determines the tree yield of the grammar (Kobele et al. 2007). 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. One possible obstacle for taking this route though is that even though regular languages are closed under linear transductions, it is still an open problem whether the derivation tree languages of an MG are, too. If they weren't, then applying a linear transduction would still yield a regular language, but not necessarily a well-formed derivation language.

At least for Focus Economy, though, closure under linear transductions may be more than we actually need. First, notice that the transducers the composition of which makes up Gen are very simple non-deterministic finite-state relabelings. Now we shouldn't expect the derivation tree languages of MGs to be closed under finite-state relabelings, despite their simplicity (the IO-context-free tree languages, for instance, aren't closed under these relabelings). However, when we consider the form of the trees in the range of Gen, it seems fairly unlikely that it couldn't be obtained directly by an MG. The only conditions are that there is a unique path of S-labeled nodes and that one node in this path is also marked for focus. Things are complicated slightly by the special status of adjuncts and anaphorically destressed nodes, but overall we are dealing with very simple conditions that a MG should have no problem with. The truly problematic question, then, is whether the tree languages of MGs are closed under intersection with a regular language, i.e. whether we can apply the filtering step to tame the overgeneration inherent to Gen. To my knowledge, this is an open problem, too, although the answer might already be implicit in the automata-theoretic perspective on MGs of Kobele et al. (2007) or the two-step approach of Kolb et al. (2003).

6 Merge-over-Move

Another well-known reference-set constraint is Chomsky's Merge-over-Move condition (MOM; Chomsky 1995b, 2000), which is the subject of inquiry in this section. After a short discussion of the mechanics of the constraint and its empirical motivation, I turn to the formal aspects of implementing MOM. In spite of the fact that MOM is what we might now — using the terminology previously introduced for OSs — call an endocentric, i.e. Fewest Steps-like constraint, the procedure for devising a MOM transducer exhibits many parallels to Focus Economy. I take this as further support of my earlier claim that both kinds of constraints can be naturally studied in the framework of OSs and tree transducers.

6.1 *Merge-over-Move Explained*

In comparison to Focus Economy, modelling MOM is slightly more intricate, because there are multiple versions of the constraint, which are seldom carefully teased apart in the literature. Naturally they all share the core idea of MOM: if at some point in a derivation we are allowed to choose between Merge and Move as the next step of the derivation, Merge is preferable to Move. This idea can be used to account for some puzzling contrasts involving expletives (if not indicated otherwise, all examples are taken from Castillo, Drury, and Grohmann 2009).

(13) a. There seems to be a man in the garden.
b. * There seems a man to be in the garden.
c. A man seems to be in the garden.

Recall that in an MG in Chomsky's sense, we start out with a multiset of lexical items — the *numeration* — that are enriched with interpretable and uninterpretable features, the latter of which have to be erased by the operation of feature checking. Under such a conception, (13a) and (13c) are easy to derive. Let us look at (13c) first. It starts out with the numeration {seems, to, be, a, man, in, the, garden}. Multiple applications of Merge yield the phrase [$_{\text{TP}}$ to be a man in the garden]]. At this point, the Extended Projection Principle (EPP) demands that the specifier of the infinitival TP be filled by some phrase. The only item left in the numeration is *seems*, which cannot be merged in SpecTP. Hence we are stuck with moving the DP *a man* into SpecTP, yielding [$_{\text{TP}}$ a man to be t$_{\text{DP}}$ in the garden]. Afterwards, the TP is merged with *seems* and the DP is once again moved, this time into the specifier of *seems* to check the case feature of the DP and satisfy the EPP.

For (13a), however, things are slightly different. Here the numeration initially consists of {there, seems, to, be, a, man, in, the, garden}. Once again we start out by merging items from the numeration until we arrive at [$_{\text{TP}}$ to be [$_{\text{DP}}$ a man in the garden]]. But now we have two options: Merger of *there*, which is later followed by moving *there* into the specifier of *seems*, thus yielding the grammatical (13a), or first moving *a man* into the specifier of *to be* and subsequently merging *there* with *seems a man to be in the garden*, which incorrectly produces the ungrammatical (13b). MOM rectifies this overgeneration problem by barring movement of *a man* into the specifier of *to be*, as the more economical route of merging *there* in this position is available to us. At the same time, MOM does not block (13c) because we aren't given a choice between Merge and Move at any point of its derivation.

Different versions of MOM emerge depending on the setting of two binary parameters:

P1 Reference set algorithm: *indiscriminate/cautious*
Indiscriminate versions of MOM (iMOM) pick the most economical derivation even if it derives an ungrammatical phrase structure tree — such derivations are said to *crash*. Cautious versions of MOM (cMOM), on the other hand, picks the most economical derivation that yields a well-formed tree.[2]

P2 Mode of application: *sequential/output*
Sequential versions of MOM (sMOM) check for MOM violations after every step of the derivation. Thus early violations of MOM carry a significantly greater penalty than later ones.[3] MOM applied to the output (oMOM), however, is sensitive only to the total number of violations, not their timing. So if derivation d incurs only one violation of MOM, which occurs at step 4 in the derivation, while derivation d' incurs seven, starting at step 5, then d will win against d' under an output filter interpretation of MOM and lose under a sequential one.

Combining the parameters in all logically possible ways (*modulo* underspecification) yields the four variants isMOM, csMOM, ioMOM and coMOM. All four of them supposedly use the *Identity of Numerations Condition* (INC) for computing reference sets, according to which the reference set of a derivation d contains all the derivations that can be built from the same numeration as d.[4] "Supposedly", because only the sMOM variants have been discussed at length in the literature. The original proposal by Chomsky (1995b) is what I call csMOM. But the global flavor of csMOM (if MOM is evaluated at every step of the derivation, i.e. before the derivation

[2]If we include crashing derivations in the reference set, however, we face the problem that the empty derivation, or a derivation that never moves anything, will always be the most grammatical option. The condition would have to be strengthened such that one may only consider derivations that obey all syntactic principles up to the first choice between Merge and Move, at which point they may later run into irreparable problems. The technical details are presumably much more complicated than that, but fortunately they are of little importance given my objectives. I will simply adopt the tentative assumption in the literature that only "reasonable" alternatives are in the reference set.

[3]This raises several thorny issues concerning the notion of derivational earliness, as derivations are usually partial rather than total strict orders. Fortunately these complications do not surface in the cases MOM was designed to account for, so I will happily ignore them. But in grammars with sidewards-movement this issue needs to be properly addressed (Drummond 2010).

[4]The astute reader may rightfully point out that the INC is both too weak and too strong. On the one hand, it erroneously allows *John likes Mary* to compete against *Mary likes John*, yet on the other hand it seems to block competition between candidates that differ only in their feature make-up, even if only marginally so. This observation is correct and highlights a problem in the specification of MOM's reference-set algorithm that has frequently been lamented in the literature (Sternefeld 1996). As it turns out, though, these formal problems, as well as certain empirical quandaries I will discuss later on, do not arise in a tree transducer model of MOM.

is completed, how does it know which competing derivations will crash later on and thus may be discarded for the comparison?) prompted the creation of isMOM as a strictly local alternative. Indeed isMOM can be argued to contain not even a modicum of reference-set computation, as it simply states that if there is a choice between Merge and Move, pick Merge. Whether such a choice exists can always be checked locally.

For simple cases like (13), where we only have to choose once between Merge and Move, all MOM variants produce the same results (although evaluation of iMOM variants is complicated by the open question which degree of deviancy one wants to allow for competing derivations; see my remarks in fn. 2). But as soon as we encounter examples involving embedded clauses, the predictions diverge (which was already noted as early as 1997 by Wilder and Gärtner).

(14) a. There was [a rumor [that a man was t_{DP} in the room]] in the air.
b. [A rumor [that there was a man in the room]] was t_{DP} in the air.

Both oMOM-versions get the right result: Each sentence prefers Move over Merge exactly once, so assuming that there are no (grammatical) competing derivations that start from the same numeration and incur fewer violations, (14a) and (14b) should both be grammatical. The sMOM variants, on the other hand, struggle with this data. The sentences are built up from the same numeration, so (14b) should block (14a), since the former violates MOM at a later derivational step than the latter. In order to account for such cases, Chomsky (2000) stratifies numerations into subnumerations such that each CP has its own numeration (which is extended in Chomsky 2001 to the contemporary phase system). In the case at hand, (14a) is built from the numeration {{there, was, a, rumor, in, the, air}, {that, was, a, man, in, the, room}}, and (14b) from the minimally different {{was, a, rumor, in, the, air}, {that, there, was, a, man, in, the, room}}. By the INC, then, derivations built from the former do not belong to the same reference set as derivations built from the latter.

So now we have a third parameter to take into account. Even though it isn't directly related to the makeup of MOM, I will indicate it as a prefix as before.

P3 Application domain: *restricted/unbounded*
Restricted versions of MOM (rMOM) are parts of a grammar where every CP has its own numeration. Unbounded versions (uMOM) belong to grammars with one big numeration.

Taking stock, we have csuMOM as the version of MOM introduced in Chomsky (1995b), isuMOM as its local counterpart, and csrMOM as the modification put forward in Chomsky (2000). Somewhat surprisingly, no oMOM variants are entertained in the literature, despite the small empirical advantage they have displayed so far. For this reason, I shall mostly restrict myself to sMOM variants in the following.

6.2 *Properties of Merge-over-Move*

Let us step back for a second to take in the architecture of MOM on a broad scale, in terms of OSs. If one abstracts away from the peculiarities of the application mode, MOM merely acts as a filter on the tree language derived by a grammar; the trees don't have to be manipulated at all, as was the case with Focus Economy. It follows that MOM is endocentric and thus output joint preserving. Moreover, reference sets presumably do not overlap, at least not if we take the identity of numerations requirement literally, whence MOM is also output partitioned. Finally, the evaluation metric is independent of the input — for oMOM, that is. With sMOM we run into a problem. Whether a step was taken earlier in comparison to other candidates cannot simply be read off the total number of violations. Instead, the evaluation itself has to proceed in a bottom up fashion, weeding out candidates with unnecessary instances of Move after every derivational step. It seems, then, that the metric involves genuine reference-set computation which requires us to consider multiple tree structures at once. However, remember that we faced a similar problem with Focus Economy, where the constraint as it was stated in the literature relied on comparing two focus-annotated trees to determine licit foci. Our answer to this problem was to represent the competing trees within a single tree (using the FocusPath and StressPath predicates) and thus emulate the comparative procedures by well-formedness constraints on this one underlying tree. If we could find a similar way of representing competing derivations within one derivation tree, a major hurdle would be out of the way.

But there is yet another problem, and this one pertains to sMOM as well as oMOM: the INC; it is impossible for a linear tree transducer to define the corresponding partition over the input language. The major culprit here is the restriction to finite memory, which entails that we can only distinguish between a bounded number of occurrences of lexical items. For some suitably large n, the multiset M' obtained from $M := \{\text{John}_n, \text{thinks}_n, \text{that}_n, \text{Mary died}\}$ by adding one more occurrence of *John*, *thinks*, and *that* will be indistinguishable from M for the transducer. Thus the challenges surrounding definability by transducers extend from the evaluation metric directly to Gen. And so does the solution.

6.3 A Model of sMOM

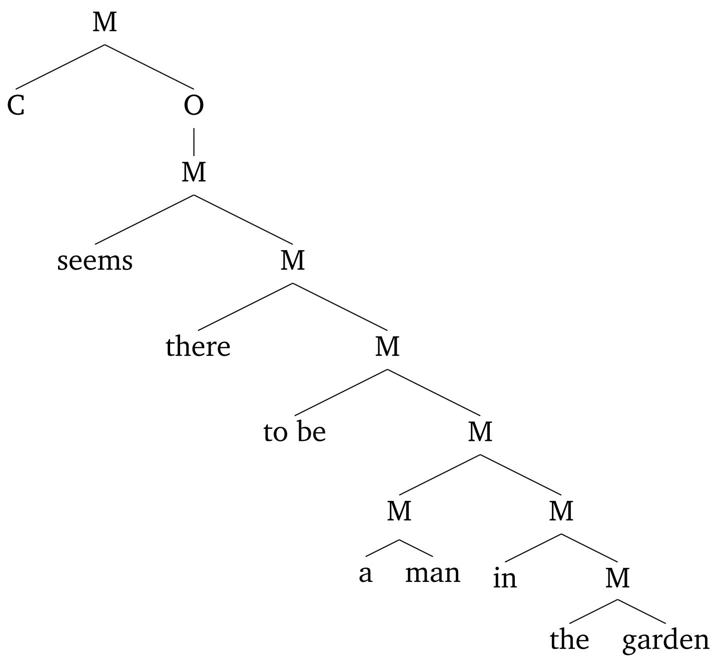
The INC is both too powerful and too weak. Consider (13) again, repeated here for the reader's convenience.

(15) a. There seems to be a man in the garden.
b. * There seems a man to be in the garden.
c. A man seems to be in the garden.

MOM's objective is to explain why (15b) is ungrammatical, and it does so by using a metric that makes it lose out against (15a). The grammaticality of (15c), on the other hand, follows from the fact that it isn't a member of the same reference set, due to the INC. But identity of numerations is a rather indirect encoding of the relationship that holds between (15a) and (15b). A formally simpler condition emerges when we look at their derivation trees (cf. Fig. 6 on page 35). Ignoring the feature specifications of the lexical items, we see that the only difference between the respective derivation trees is the timing of move. Rather than a transducer modelling the INC, then, all we need is a transducer that will produce (at least) the derivation trees for (15a) and (15b) when given either as an input. This involves merely changing the position of the unary branch, which is an easy task for a linear transducer. But now compare these derivations to the one for (15c) in Fig. 7 on page 36. The derivation trees of (15a) and (15b) are essentially the result of non-deterministically replacing one instance of move in the derivation tree of (15c) by merger with expletive *there*. Strikingly, though, rewriting the lower occurrence of O yields the grammatical (15a), whereas rewriting the structurally higher occurrence gives rise to the ungrammatical (15b). Now if we design the transducer such that it won't rewrite an O as a *there*-merger after it has already passed up on an opportunity to do so earlier in the derivation, (15b) cannot be generated from the derivation tree of (15c). In linguistic parlance, this is tantamount to treating MOM as a well-formedness condition on derivation trees (note the similarity to the Focus Economy strategy).

The idea just outlined is captured as follows: First, we take as our input language I the set of derivation trees of an MG that generates the intended derivation trees. Then, we use a transducer α to map this language to a set U of underspecified derivation trees. The transducer strips away all features from the lexical items, deletes expletive *there* and rewrites O as O/there in the TP-domain. The underspecified representation of (15a)–(15c), for instance, is almost identical to the derivation tree of (15c) except that the two O nodes are now labeled O/there (and the lexical items are devoid of any features). These underspecified representations are then turned into fully specified representations again by the transducer β. It reinstantiates the features on the lexical items and non-deterministically rewrites O/there as O or Merger of *there*, but with the added condition that once an O/there node has been replaced by an O, all remaining instance of O/there in the same CP have to be rewritten as O. I call the output language of the second transduction J. The name is meant to be a shorthand for *junk*, as J will contain a lot thereof, for two independent reasons. First, the non-deterministic rewriting of O/there allows for two occurrences of O/there to be rewritten as *there*, which yields the (derivation tree of the) ungrammatical *there seems there to be a man in the garden*. Second, the reinstantiation of the features is a one-to-many map that will produce a plethora of illicit derivation trees as some lexical items may not be able to get all their features checked. This overgeneration problem is taken care of by intersecting J with I. The overall structure of the computation, in comparison to Focus economy and the under specification strategy in general, is depicted in Fig. 8 on page 36. Note that in terms of OSs, the actual content of MOM is now squeezed into Gen and the only constraint of the OS is the input language itself.

Just as with Focus Economy, there is some reason for concern as it is still an open question whether Minimalist derivation tree languages are closed under intersection with regular languages (as regular sets are closed under linear transductions, J is guaranteed to be regular, so we only need to know whether $I \cap J$ is a derivation tree language for some MG). For sMOM itself I expect that $I \cap J$ is the derivation tree language of some MG, as it only removes a few derivation trees from I, which could also be achieved by judicious use of the feature calculus.



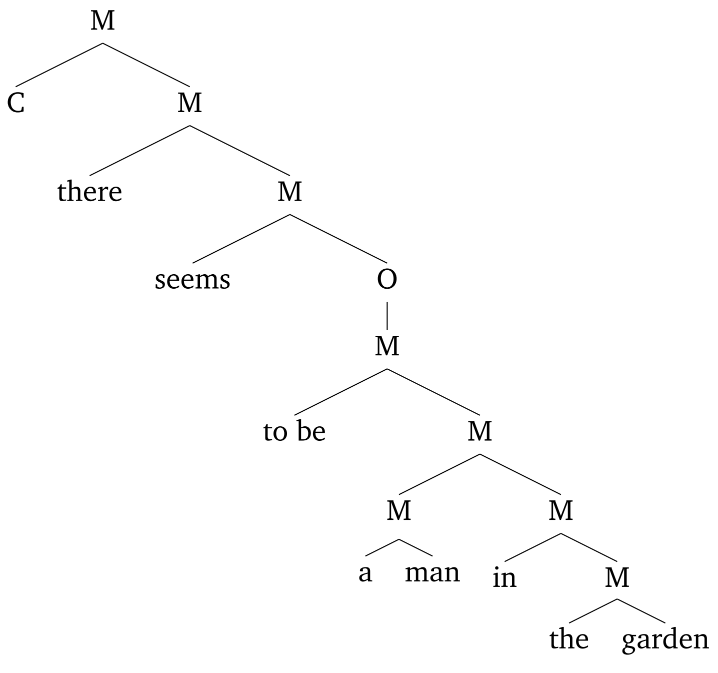


Figure 6: The derivation trees of (15a) and (15b) differ only in the position of the unary branch

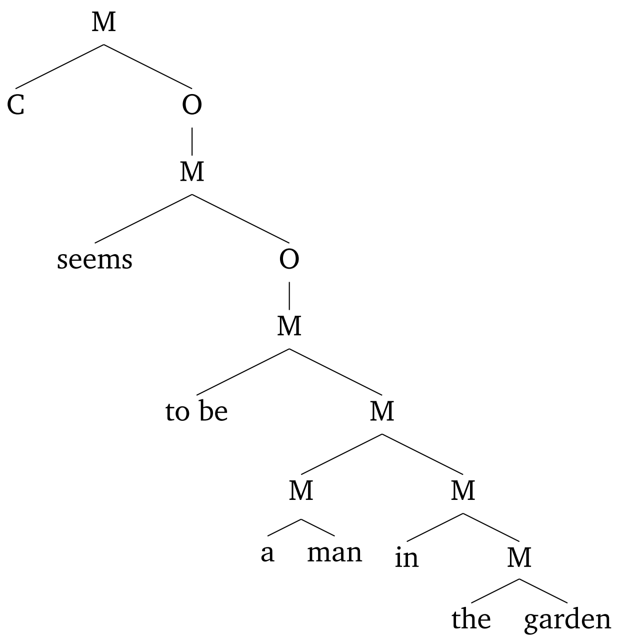


Figure 7: The derivation tree of (15c) can be taken as a basis for the previous two

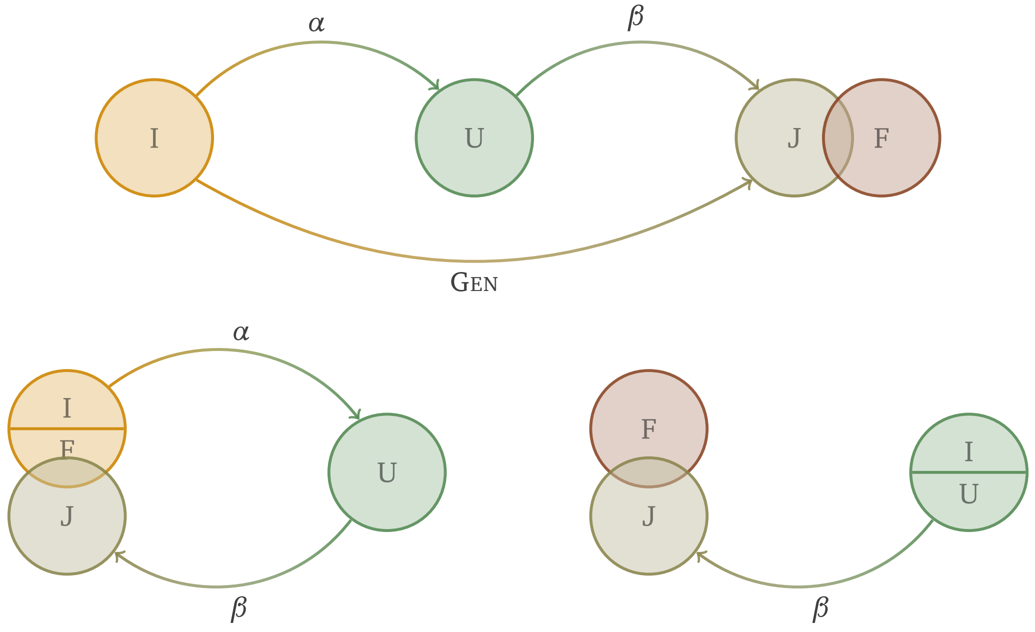


Figure 8: Architecture of the underspecification-and-filtration strategy in general (top), sMOM (left) and Focus Economy (right) in comparison

After these important remarks, let us get started on the low-level implementation of MOM. As mentioned before, I assume that I is given by some MG $\mathscr{E} := \langle \Sigma_{\mathscr{E}}, F_{\mathscr{E}}, \mathit{Types}, \mathit{Lex}_{\mathscr{E}}, O \rangle$. The specifics of $\mathscr{E}$ will be elaborated in the next section, for now I only have to assume that the derivation trees are fully labeled such that leave nodes are decorated by items drawn from $\mathit{Lex}_{\mathscr{E}}$ and unary and binary branching nodes, respectively, by M and O (short for *merge* and *move*). The transduction α is obtained from composing the two transducers *Remove Features* and *Underspecify*.

Definition 36. *Remove Features* is the deterministic (one-state) relabeling that maps each $l := \langle \sigma :: f_1, \ldots, f_{base}, \ldots, f_n \rangle \in \mathit{Lex}_{\mathscr{E}}$ to $l' := \sigma_{f_{base}}$, where f_{base} is the base feature of l. The set of these simplified lexical items is denoted by Λ.

Even though the definition of an MG in Sec. 1 allows for a lexical item to have several base features or none at all, neither option is ever exploited for real-life grammars, so *Remove Features* is well-defined. If for some reason multiple base features (or their absence) are indispensable, the transduction can be extended such that each lexical item is subscripted by the n-tuple of its n base features. In either case, the map defined by *Remove Features* is many-to-one, so Λ is finite by virtue of the finiteness of $\mathit{Lex}_{\mathscr{E}}$.

Definition 37. *Underspecify* is the lbutt $\mathscr{U}$, where $\Sigma_{\mathscr{U}} := \Lambda \cup \{M, O\}$, $\Omega_{\mathscr{U}} := \Sigma_{\mathscr{U}} \cup \{\text{O/there}\}$, $Q := \{q_*, q_c, q_i, q_t\}$, $Q' := \{q_*\}$, and $\Delta_{\mathscr{U}}$ consists of the rules below, where I use the following notational conventions:

- σ_I (σ_C) denotes any lexical item $l \in \Lambda$ whose base feature is I (C),
- the symbol "there" refers to any expletive $l \in \Lambda$ involved in MOM (usually just *there*, but possibly also *it*),
- σ_l denotes any lexical item which doesn't fall into (at least) one of the categories described above,
- as derivation trees aren't linearly ordered, rules for binary branching nodes are given for only one of the two possible orders (namely the one that reflects the linear order in the derived structure).

$$\begin{array}{rr}
\sigma_l \to q_*(\sigma_l) & \mathrm{M}(q_{c*}(x), q_{i*}(y)) \to q_*(\mathrm{M}(x,y)) \\
\sigma_I \to q_i(\sigma_I) & \mathrm{M}(q_i(x), q_{\{i,*\}}(y)) \to q_i(\mathrm{M}(x,y)) \\
\text{there} \to q_t(\text{there}) & \mathrm{M}(q_t(x), q_{\{i,*\}}(y)) \to q_i(\text{O/there}(y)) \\
\sigma_C \to q_c(\sigma_C) & \mathrm{O}(q_*(x)) \to q_*(\mathrm{O}(x)) \\
 & \mathrm{O}(q_i(x)) \to q_i(\text{O/there}(x))
\end{array}$$

The underspecified derivation have to be turned back into fully specified ones by the transduction β, which is the composition of *Path Condition* and the inverse of *Remove Features*.

Definition 38. *Path Condition* is the lbutt $\mathscr{P}$, where $\Sigma_{\mathscr{P}} := \Omega_{\mathscr{U}}$, $\Omega_{\mathscr{P}} := \Sigma_{\mathscr{U}}$, $Q := \{q_*, q_c, q_o\}$, $Q' := \{q_*\}$, and $\Delta_{\mathscr{P}}$ contains the rules below (the same notational conventions apply):

$$\begin{aligned}
\sigma_l &\to q_*(\sigma_l) & \mathrm{M}(q_{c\{c,*\}}(x), q_{o\{c,*\}}(y)) &\to q_*(M(x,y))\\
\sigma_I &\to q_*(\sigma_I) & \mathrm{M}(q_{\{*,o\}}(x), q_o(y)) &\to q_o(\mathrm{M}(x,y))\\
\sigma_C &\to q_c(\sigma_C) & \mathrm{O}(q_*(x)) &\to q_*(\mathrm{O}(x))\\
&& \mathrm{O/there}(q_*(x)) &\to q_*(\mathrm{M}(\mathrm{there}, x))\\
&& \mathrm{O/there}(q_{\{o,*\}}(x)) &\to q_o(\mathrm{O}(x))
\end{aligned}$$

The crucial step toward capturing MOM is the last rule of *Underspecify*, which tells the transducer that after it has rewritten one instance of O/there as O, it has to switch into state q_o, which tells it to always rewrite O/there as O. Only if it encounters a node of category C may the transducer switch back into its normal state q_* again.

We can alternatively restrict α to *Remove Features*, express the composition of *Underspecify* and *Path Condition* as an MSO constraint ψ on the surface language of *Remove Features* applied to the derivation language of $\mathscr{E}$ (since said language is regular), and let β be the inverse of α. All ψ has to do is pick out the path of nodes from a leaf of category I to the closest dominating node that is the mother of a node of category C and stipulate that no node in this path may be both the mother of an expletive and dominate an O-node belonging to the same path. This can easily be made precise.

$$\mathrm{Start}(X,x) \leftrightarrow X(x) \wedge \neg\exists y[X(y) \wedge \neg(x \approx y) \wedge y \triangleleft^* x]$$

$$\mathrm{End}(X,x) \leftrightarrow X(x) \wedge \neg\exists y[X(y) \wedge \neg(x \approx y) \wedge x \triangleleft^* y]$$

$$\begin{aligned}
\mathrm{IPdomain}(X) \leftrightarrow{}& \exists! x[\mathrm{Start}(X,x)] \wedge \exists! x[\mathrm{End}(X,x)] \wedge \forall x[\mathrm{End}(X,x) \to I(x)] \wedge\\
& \forall x\Big[\mathrm{Start}(X,x) \to \exists y\left[x \triangleleft y \wedge C(y) \wedge \neg\exists z[y \triangleleft z]\right]\Big] \wedge\\
& \forall x,y,z[X(x) \wedge X(y) \to (x \triangleleft^* y \vee y \triangleleft^* x) \wedge (\neg X(z) \to \neg(x \triangleleft z \wedge z \triangleleft y)]
\end{aligned}$$

Here the predicates I and C pick out the same lexical items as σ_I and σ_C did before. Similarly, I use the predicate Exp in the statement of ψ to denote expletives. The path condition on derivation trees is then approximated by ψ as follows.

$$\psi := \forall X\Big[\mathrm{IPdomain}(X) \to \neg\exists x\left[X(x) \wedge \exists y[x \triangleleft y \wedge \mathrm{Exp}(y)] \wedge \exists z[X(z) \wedge O(z) \wedge x \triangleleft^* z]\right]\Big]$$

As in the case of Focus Economy, ψ by itself does not fully capture the constraint, but over the set of derivation trees of the MG $\mathscr{E}$ it does.

The original transducer architecture (depicted in Fig. 8) differs from the revised version (Fig. 9) in that while both yield the same output language, only the former properly captures the relation between trees established by MOM. In the alternative, no trees of the input are ever related to each other, there is no reference-set algorithm to speak of; instead, it simply enforces a well-formedness condition on derivation

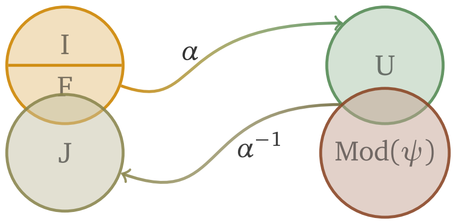


Figure 9: A different perspective on sMOM

trees. In fact, the removal and reinstantiation of features is redundant in this approach. If we proceed as I proposed originally, on the other hand, the result is a relation that will group trees into reference-sets and for each tree in reference-set R return the optimal trees in R as its outputs. One might say that both models capture the *weak relational capacity* of MOM insofar as they yield the correct output language, but only the more elaborate model of Fig. 8 faithfully represents MOM's *strong relational capacity*, the actual transduction.

6.4 *Empirical Evaluation*

As discussed above, the transducer model of MOM accounts for simple expletive/ non-expletive alternations as in (15). Instead of going through another iteration of the same basic argument, let us look at a more complex example that we have encountered before, repeated here as (16).

(16) a. There was [a rumor [that a man was t_{DP} in the room]] in the air.
b. [A rumor [that there was a man in the room]] was t_{DP} in the air.

Recall that this was a problematic case for pre-Chomsky (2000) versions of MOM (i.e. csuMOM and isuMOM), because in the absence of stratified numerations the INC puts (16a) and (16b) in the same reference set, where they have to compete against each other. Under a sequential construal of MOM, then, (16a) will block (16b) as it opts for Merge rather than Move at the first opportunity.

Under the transducer conception of MOM (tMOM), on the other hand, (16) is a straightforward generalization of the pattern in (15). The underspecified derivation tree of both sentences is shown in Fig.10. When the underspecified derivation is expanded to full derivations again, all four logical possibilities are available: *there*-insertion in both CPs, Move in both CPs, *there*-insertion in the lower CP and Move in the higher one, and Move in the lower CP and *there*-insertion in the higher one. The last option is available because the transducer, which is in the "rewrite all instances of O/there as O"-state q_o after rewriting the label O/there as O, switches back into the neutral state q_* after encountering the CP headed by *that*. Thus when it encounters the second O/there node in the higher CP, it can once again choose freely how to rewrite it. Provided the four derivation trees obtained from the underspecified derivation aren't filtered out by the MG, they are in turn transformed into the following derived structures, all of which are grammatical:

(17) a. There was a rumor that there was a man in the room in the air.

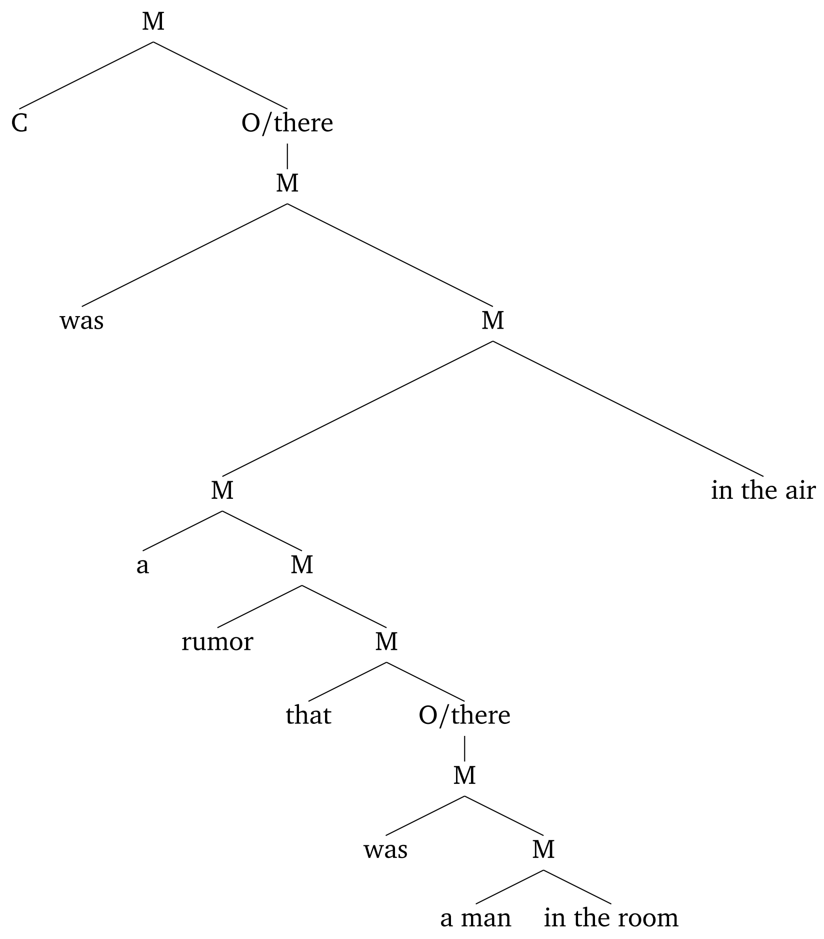


Figure 10: Underspecified Derivation Tree of (16a) and (16b)

b. There was a rumor that [a man]$_i$ was t_i in the room in the air.
c. [A rumor that there was a man in the room]$_i$ was t_i in the air.
d. [A rumor that [a man]$_i$ was t_i in the room]$_j$ was t_j in the air.

The identity of numerations condition of the original version of MOM entails that these four sentences belong to three distinct equivalence classes, one containing (17a), one containing (17b) and (17c), and one containing (17d). MOM enriched with stratified numerations, on the other hand, puts each sentence into its own equivalence class. Only tMOM lumps them all together into one equivalence class, which is the more plausible route to take, at least intuitively.

The very fact that Merge variants as well as Move variants can be obtained from the same underspecified derivation indicates that the transducer version is less of a relativized ban against Merge and more of a description of the set of possible continuations of a derivation once a choice pro-Merge or pro-Move has been made. This idea is actually what underlies the restatement of the transducer *Path Condition* in MSO terms by the constraint ψ. Empirically, this has the welcome effect that we do not run into the undergeneration problems that plague isMOM, and to a lesser degree csMOM. Consider the following utterances.

(18) a. It seems that John was in the room.
b. * John seems it was in the room.

The derivation for either sentence starts out by assembling the small clause [John [in the room]], which is subsequently merged with a T head (phonetically realized by *was*). Now isMOM would enforce base-merger of *it* into the specifier of the TP, rather than movement of *John* into said position. From there on, only ungrammatical structures can be generated. Either *John* remains in situ and the derivation crashes because of the unchecked case feature of *John*, or *John* moves over the expletive into SpecTP of the matrix clause, in violation of the Shortest Move Condition. The only grammatical alternative, (18a), cannot be generated because it is blocked by isMOM. With tMOM one does not run into this problem, as it will generate both sentences, but the second one will probably be filtered out by the MG itself because of the illicit movement step. The csMOM variant alternates between the two options: If (18b) is ungrammatical for independent reasons, (18a) does not have to compete against it and will emerge as the winner, just as with the transducer model. If (18b) is grammatical, it will block (18a), in line with isMOM.

This general theme is repeated in various configurations where other versions of MOM undergenerate. Shima (2000) lists a number of cases where Merge-over-Move makes false predictions and, in fact, something along the lines of a Move-over-Merge principle seems to be required.

(19) a. It is asked [how likely t_{John} to win]$_i$ John is t_i.
b. * John is asked [how likely t_{John} to win]$_i$ it is t_i.

The assembly of [is [how likely John to win]] proceeds as usual. At this point, a decision has to be made as to whether we want to move *John* into SpecTP or base-merge the expletive instead. The isMOM variant once again picks the base-merger route, so we end up with [it [is [how likely John to win]]. After this phrase is merged with *asked* and *is*, *John* moves into the specifier of the matrix TP to get its case feature checked. Unless moving *John* is barred for independent reasons, (19b) will be grammatical, so that (19a) will be blocked under both indiscriminate and cautious construals of MOM. Thus we get the following contrast between different versions of MOM. The variant isMOM always blocks (19a), csMOM blocks it only if (19b) is grammatical, and tMOM never blocks it. So for both csMOM and tMOM we have to make sure that our MG $\mathscr{E}$ contains some locality condition that rules out (19b). A natural candidate would of course be the islandhood of [how likely John to win].

We also have to make further assumptions about $\mathscr{E}$ to rule out cases of superraising like (20a) and multiple occurrences of *there* as in (20b). On a conceptual level, this is a defensible move as the deviancy of those examples does not seem to be directly related to MOM, and they are hardly ever discussed with respect to MOM in the literature. However, if we really wanted to incorporate those restrictions into MOM, at least the ban against double *there* can easily be accommodated by changing from a "once you go O, you never go back" version of *Path Condition* to "once you choose, it's always O". This is easily accomplished by replacing the rule $\mathrm{O/there}(q_*(x)) \to q_*(\mathrm{M}(\mathrm{there}, x))$ by the minimally different $\mathrm{O/there}(q_*(x)) \to q_o(\mathrm{M}(\mathrm{there}, x))$.

(20) a. * A man seems there to be in the room.
  b. * There seems there to be a man in the room.

Interestingly, at least German allows for multiple expletives to occur in a single clause, even within the *mittelfeld*, which is usually considered a part of the TP. Examples are given in (21) (my own judgments). As multiple expletives can be hosted by German TPs, the contrast between German and English can't be reduced to the fact that German mandatorily requires SpecCP to be filled and thus has two specifiers that may host expletives.

(21) a.

Es/Da	scheint	da	ein	Mann	im	Garten	zu	sein.
it/there	seems	there	a	man	in.the	garden	to	be

b.

Es/?Da	scheint	da	ein	Mann	da	im	Garten	zu	sein.
it/there	seems	there	a	man	there	in.the	garden	to	be

'There seems to be a man in the garden.'

c.

Es/?Da	scheint	da	ein	Mann	im	Garten	da	zu	sein.
it/there	seems	there	a	man	in.the	garden	there	to	be

'There seems to be a man in the garden.'

If we assume that economy principles are universal, then any cross-linguistic variation has to arise from other grammar-internal factors. From a transducer perspective, though, there are no good reasons for such a stipulation. As long as language-specific variants of a constraint all belong to the same transducer class, they are all equally economic in a mathematical sense. In the case of *Path Condition*, the slight modification proposed above has absolutely no effect on the runtime-behavior of the transducer, nor is it in any tangible way less intuitive or less "Minimalist". Reference-set constraints must not be artificially kept away from matters of crosslinguistic variation, because this is an empirical domain where they are in principle superior to standard well-formedness conditions. This has not been noticed in the syntactic literature yet — e.g. for Müller and Sternefeld (1996:491) "it is [...] not clear how a [reference-set; TG] constraint like Economy can be rendered subject to parametrization" — but in contrast to well-formedness conditions these constraints offer multiple loci of parametrization: the transductions α and β, and the definition of the filter as well as at which point of the transduction it is applied. Now that our formal understanding of reference-set constraints has finally reached a level where at least such basic questions can be given satisfactory answers, the initial empirical questions can be reapproached from a new angle that challenges the received wisdom on when, where and how reference-set constraints should be employed.

7 Shortest Derivation Principle

In the last section, I left open how to formalize oMOM, the variant of MOM which doesn't weigh violations depending on how early they happen in the derivation. In other words, oMOM simply counts the number of violations and picks the candidate(s) that incurred the least number of violations. This is very close in spirit to the *Shortest Derivation Principle* (SDP) of Chomsky (1991, 1995a), which I (and many authors before me) have also referred to as *Fewest Steps*.[5] The SDP states that if two convergent (i.e. grammatically well-formed) derivations are built from the same lexical items, pick the one with the fewest operations. Usually, the set of operations considered by the economy metric is assumed to comprise only Move, the reason being that Merge is indispensable if all the lexical items are to be combined into a single phrase marker. Naively, then, oMOM is but a variant of the SDP that doesn't penalize every instance of Move but only those where Merge would have been a feasible alternative. Even though I will refrain from discussing oMOM any further in this section and focus on the SDP instead, their close relation means that after reading this and the previous section, the reader will be in possession of all the tools required to formalize oMOM. In fact, I explicitly encourage the reader to draw at least a sketch of the implementation to test their own understanding of the material.

Returning to the SDP, I will explore two variants of this principle, one being the original proposal and the other one the result of extending the set of operations that enter the economy metric to Merger of phonologically unrealized material such as (certain) functional heads in the left periphery. The underlying intuition of this extension is that covert material should be merged only if is required for convergence. This would certainly be close in spirit to GB-analyses of English where main clauses are TPs unless a CP is required as a landing site for wh-movement or possibly QR. Curiously, this *prima facie* innocent modification has the potential to push the SDP out of the realm of linear transductions: the SDP restricted to Move can be defined by a linear transducer, whereas the SPD applied to Move and Merge of covert material is not, unless restrictions on the distribution of silent heads are put into place.

[5]Technically, Fewest Steps is the original formulation and the SDP its more "minimalist" reformulation that does away with representational machinery such as Form-Chain. This makes it a better fit for MGs in the sense of Stabler and Keenan (2003), and for this reason I prefer the name SDP over the better-known Fewest Steps.

7.1 *The Shortest Derivation Principle Explained*

To give the reader a better feeling for the constraint I do as in the previous sections and present a simple example first. It is a well-known fact that A-movement in English exhibits freezing effects. While arguments may be extracted from a DP in complement position, as soon as the DP A-moves to a higher position, usually SpecTP, extraction is illicit — the DP's arguments are frozen in place. This contrast is illustrated in (22).

(22) a. Who$_i$ did John take [$_{DP}$ a picture of t_i]?
b. * Who$_i$ was [$_{DP_j}$ a picture of t_i] taken t_j by John?

At first (22b) seems to be a mere instance of a CED-effect (Huang 1982) as in (23), so whatever rules out the latter should also take care of (22b).

(23) * Who$_i$ is [$_{DP}$ a picture of t_i] on sale?

The corresponding derivation for this analysis of the ungrammaticality of (22b) would be (24).

(24) a. [$_{VP}$ taken [$_{DP_j}$ a picture of who$_i$] by John]
b. [$_{TP}$ [$_{DP_j}$ a picture of who$_i$] T [$_{VP}$ taken t_j by John]]
c. [$_{CP}$ who$_i$ was [$_{TP}$ [$_{DP_j}$ a picture of t_i] T [$_{VP}$ taken t_j by John]]]

Notably, though, the DP in (22b) is not base-generated in subject position but in object position, so in theory it should be possible to extract the wh-word from the DP before it moves into subject position and thus becomes a barrier for movement in the sense of Chomsky (1986). There are two distinct derivations that make use of this loophole, the relevant stages of which are depicted in (25) and (26) below.

(25) a. [$_{VP}$ taken [$_{DP_j}$ a picture of who$_i$] by John]
b. [$_{CP}$ who$_i$ was [$_{TP}$ T [$_{VP}$ taken [$_{DP_j}$ a picture of t_i] by John]]]
c. [$_{CP}$ who$_i$ was [$_{TP}$ [$_{DP_j}$ a picture of t_i] T [$_{VP}$ taken t_j by John]]]

(26) a. [$_{VP}$ taken [$_{DP_j}$ a picture of who$_i$] by John]
b. [$_{VP}$ who$_i$ taken [$_{DP_j}$ a picture of t_i] by John]
c. [$_{TP}$ [$_{DP_j}$ a picture of t_i] T [$_{VP}$ who$_i$ taken t_j by John]]
d. [$_{CP}$ who$_i$ was [$_{TP}$ [$_{DP_j}$ a picture of t_i] T [$_{VP}$ taken t_j by John]]]

The first derivation can be ruled out on grounds of the extension condition, which bans countercyclic movement. The second, however, seems to be well-formed, provided that extraction of the wh-phrase is licensed by feature checking (in a phase-based approach, this could be handled by an EPP/OCC-feature, for instance). So we erroneously predict that (22b) should be grammatical.

Collins (1994) solves this puzzle by recourse to the SDP. Note that (24) and (25) involve one movement step less than (26). So if (26) has to compete against at least one of the two, it will be filtered out by the SDP. The filtering of (24) and (25), respectively, is then left to the subject island constraint and the ban against countercyclic movement (whatever their technical implementation might be in our grammar).

The reader may be wondering why non-convergent derivations are suddenly parts of the reference-sets. From a formal perspective, there is little reason to argue against this shift in perspective; as we will see in a second, it changes nothing about our general procedure. However, from a linguistic point of view this point is definitely worth elaborating. The answer is that earlier in this section I allowed myself a small degree of sloppiness when explaining convergence, as the derivations above are in fact convergent. For even though all well-formed derivations are by definition convergent, the latter is not true in general. Convergence means that all items from the numeration were merged correctly and that no feature was left unchecked, while well-formedness also implies that no other constraints were violated.

7.2 A Model of the Shortest Derivation Principle

The SDP features interesting extensions of the constraints seen so far. As it punishes every single instance of Move, it is indeed a counting constraint, in contrast to Focus Economy and MOM, which relied on surprisingly simple well-formedness conditions on somewhat peculiar paths. Moreover, it involves two cycles of under-specification and filtration rather than one, and those two cycles furthermore happen to interlock in a non-trivial way. Nonetheless the formalized version of SDP actually uses the simplest transducers of all in this paper, so it should be easy to fathom for everyone who has already mastered Focus Economy and MOM.

As with MOM, we start out with the set of derivation trees of some MG $\mathcal{E}$. And as we did before, we immediately strip away all the features except the category feature, which is preserved so that distinct trees with identical string components won't be put in the same reference set later on.

Definition 39. *Remove Features* is the deterministic (one-state) relabeling that maps each $l := \langle \sigma :: f_1, \ldots, f_{base}, \ldots, f_n \rangle \in Lex_{\mathcal{E}}$ to $l' := \sigma_{f_{base}}$, where f_{base} is the base feature of l. The set of these simplified lexical items is denoted by Λ.

In the next step, we have to ensure that two derivations wind up in the same reference set if and only if they differ merely in their number of movement steps. To this end, we first define a transducer that deletes all unary branches (i.e. branches representing Move) from the derivation, and then another one which arbitrarily reinserts unary branches. This will generate derivations that were not present at the stage immediately before we removed all instances of Move, but as the reader might have guessed, this can easily be fixed by following our own example set in the previous section and use the input language as a filter — only this time the "input language" isn't the derivation language of $\mathcal{E}$ but the language serving as the input to the transducer that removed all movement nodes, i.e. the output language of *Remove Features*. The result of these three transductions and the filtration is a transduction that relates only those (feature-free) derivations that are built from the same lexical items.

Definition 40. *Remove O* is the deterministic ltdtt $\mathscr{R}$, where $\Sigma_{\mathscr{R}} := \Lambda \cup \{M, O\}, \Omega := \Sigma_{\mathscr{R}} \setminus \{O\}, Q = Q' := \{q\}$, and $\Delta_{\mathscr{R}}$ consists of the rules below:

$$\sigma \to q(\sigma) \qquad M(q(x), q(y)) \to q(M(x,y))$$
$$O(q(x)) \to q(x)$$

Definition 41. *Insert O* is the non-deterministic ltdtt $\mathscr{I}$, where $\Sigma_{\mathscr{I}} := \Omega_{\mathscr{R}}$, $\Omega_{\mathscr{I}} := \Sigma_{\mathscr{R}}$, $Q = Q' := \{q\}$, and $\Delta_{\mathscr{I}}$ contains the rules below, with $O^{\leq n}$ denoting n-many unary O-labeled branches or less for some fixed, non-negative n:

$$\sigma \to q(\sigma) \qquad M(q(x), q(y)) \to O^{\leq n}(M(x,y))$$

One strong restriction of *Insert O* is that at any node in the derivation tree it can only insert a finite number of movement steps. This is so because a transducer may only have finitely many rules and after every step in the transduction the transducer has to move one step up in the input tree, so it cannot remain stationery at one point and keep inserting one unary branch after another until it finally decides to move on. A transducer with such capabilities is said to have ϵ-moves, and such transducers do not share the neat properties of their standard brethren. However, the restriction to only finitely many unary branches per rewrite-step is immaterial for MGs. This follows from the simple observation that since the number of features per lexical item is finite, and so is the lexicon itself, there is a longest string of features for each grammar. The length of this string dictates how many movement steps may be licensed by a single lexical item, and thus there is an upper bound on the number of movement steps between any two instances of Merge. If we are dealing with more specialized types of Move such as Sidewards Movement (Nunes 2004; Drummond 2010), countercyclic movement or asymmetric checking, the argument may not go through without further assumptions, but for a canonical MG, the limits of the transducer are inconsequential because they are also limits of the grammar.[6]

The composition of *Remove Features*, *Remove O* and *Insert O* will be our reference-set algorithm. The next step, then, is the definition of the economy metric. But for this not much more work is needed, because the metric is already given by *Insert O*. The transducer all by itself already defines rel_1^{GEN} (see Def. 10 on page 11), the ranking of all output candidates relativized to those candidates that compete against each other, so all we have to do is follow Jäger's procedure as outlined in Sec. 2. Recall the basic intuition: the transducer defines a relation $<$ on the output candidates such that $o < o'$ iff o' is the result of applying the transducer to o. Given this relation, a few nifty regular operations are enough to filter out all elements that are not minimal with respect to $<$, i.e. the suboptimal candidates. The result will be a transduction mapping, as desired, inputs to the derivation tree(s) over the same lexical items that contain(s) the fewest instances of Move — it only remains for us to reinstantiate the features, which is taken care of by the inverse of *Remove Features*.

The overall architecture of the SDP model is depicted in Fig. 11 on the facing page. Switching back for a second into the OS perspective, it is also easy to see that

[6]That countercyclic movement poses a challenge is somewhat unsatisfying insofar as it surfaces in Collins's analysis of (25). Fortunately, though, his general argument goes through even if only (25) is taken into account.

both type-level optimality and output joint preservation are satisfied (if it weren't for the removal of features, the OS would even be endocentric), thereby jointly implying global optimality.

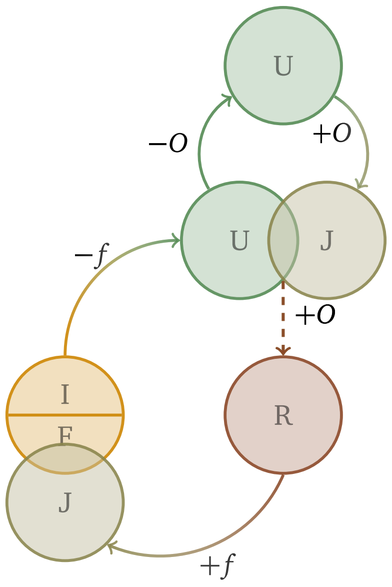


Figure 11: Architecture of the SDP

The astute reader may point out that my implementation of the SDP, while technically correct, leads to both underapplication and overapplication of the intended principle. Overapplication is caused by the indiscriminate removal of features, in particular movement-licensors as are involved in topicalization, wh-movement and (possibly) scrambling. As a consequence, these kinds of movement will appear redundant to the SDP and lose out to the derivation that involves only standard A-movement. This is easily fixed by "blacklisting" these features such that they have to be preserved by *Remove Features*.

Underapplication, on the other hand, is due to the lack of a transduction that would remove covert material whose only purpose is to host a movement-licensor feature. So if, say, a topicalization feature is always introduced by the category *Topic* à la Rizzi (1997, 2004), a derivation hosting this functional element will never compete against a derivation without it. For topicalization, this is actually a welcome result and presents an alternative for avoiding overapplication. In general, though, this must be regarded as a loophole in the SDP that needs to be fixed lest the principle can be deprived of any content by assigning every movement feature its own functional category. A solution is readily at hand: Extend *Remove O* and *Insert O* such that they may also remove or insert certain functional elements, just like MOM's *Underspecify* may remove instances of expletive *there* that can later be reinserted by *Path Condition*.

While the parametrization of *Remove Features* poses no further problems irrespective of the MG involved, extending *Remove O* and *Insert O* to functional categories will produce the correct results only if our initial grammar does not allow for recursion in the set of categories that the transducer should remove. In other words, there has to be an upper limit on the number of removable categories that can be merged subsequently before non-removable material has to be merged again. This is because of the previously mentioned inability of linear transducers to insert material of unbounded size. On a linguistic level, the ban against recursion in the functional domain is fairly innocent as even highly articulated cartographic approaches give rise only to a finite hierarchy of projections.

Conclusion

I showed that despite claims to the contrary, reference-set constraints aren't computationally intractable — in fact, many of them do not even increase the expressivity of the underlying grammar. The route towards this result was rather indirect. I first introduced controlled OSs as a formal model for reference-set constraints, focusing on the subclass of output joint preserving OSs, which is general enough to accommodate most reference-set constraints. For this class I then gave a new characterization of global optimality and used it to argue that in general, reference-set constraints are globally optimal. The shift in perspective induced by controlled OSs also made it apparent that out of the other four conditions which together with global optimality jointly guarantee that an OS stays within the limits of linear tree transductions, two 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. This highlights how surprisingly restricted reference-set constraints are in comparison to optimality systems, even though the latter seem to struggle with reference-set like conditions such as output-output correspondence (Benua 1997; Potts and Pullum 2002).

In order to demonstrate that Gen and the evaluation metric do not pose a problem either, I exhibited formally explicit implementations of three different reference-set constraints: Focus Economy, Merge-over-Move, and the Shortest Derivation Principle. The general approach followed a strategy of underspecification-and-filtration (Fig. 8) based on the following insights:

- A reference-set algorithm is likely to be computable by a linear transducer if there is a data-structure (e.g. derivation trees) such that all members of a reference-set can be uniquely described by this structure.
- Neither the mapping from inputs to underspecified structures nor the one from underspecified structures to output candidates may require insertion of material of unbounded size.
- The economy metric may be implemented as a well-formedness condition on underspecified structures, an instruction for how to turn those structures into output candidates, or a relation over underspecified structures computable by a linear transducer.

My estimate so far is that all reference-set constraints are compatible with the underspecification strategy, and that all syntactic reference-set constraints also adhere to condition 2. Combined with the positive results obtained in this paper, this suggest that reference-set constraints are significantly better behaved than is usually believed. The overall picture that emerges is that of reference-set constraints as an unexpectedly undemanding kind of linguistic constraint.

One important issue had to be left open, though, namely the closure properties of Minimalist surface tree and derivation tree languages, in particular with respect to linear transductions and intersection with regular tree languages. Even though it has no immediate bearing on the formal models in this paper (the constraints can be emulated by conditions on the distribution of features in an MG), a lack of closure under these operations, in particular closure of Minimalist derivation tree languages under intersection with regular sets, would likely prove a severe impediment to the applicability of the underspecification-and-filtration strategy. But even then everything wouldn't be lost since closure under intersection with any regular language is arguably a more general property than what we actually need.

Acknowledgments

I am greatly indebted to Ed Stabler and Uwe Mönnich as well as the two anonymous FG2010-reviewers for their motivational comments and helpful criticism. The research reported herein was supported by a DOC-fellowship of the Austrian Academy of Sciences.

References

Aoun, Joseph, Lina Choueiri, and Norbert Hornstein. 2001. Resumption, movement and derivational economy. *Linguistic Inquiry* 32:371–403.

Benua, L. 1997. Transderivational identity: Phonological relations between words. Doctoral Dissertation, UMass.

Castillo, Juan Carlos, John E. Drury, and Kleanthes K. Grohmann. 2009. Merge over move and the extended projection principle: MOM and the EPP revisited. *Iberia* 1:53–114.

Chomsky, Noam. 1986. *Barriers*. Cambridge, Mass.: MIT Press.

Chomsky, Noam. 1991. Some notes on economy of derivation and representation. In *Principles and parameters in comparative grammar*, ed. Robert Freidin, 417–454. Cambridge, Mass.: MIT Press.

Chomsky, Noam. 1995a. Categories and transformations. In *The minimalist program*, chapter 4, 219–394. Cambridge, Mass.: MIT Press.

Chomsky, Noam. 1995b. *The minimalist program*. Cambridge, Mass.: MIT Press.

Chomsky, Noam. 2000. Minimalist inquiries: The framework. In *Step by step: Essays on minimalist syntax in honor of Howard Lasnik*, ed. Roger Martin, David Michaels, and Juan Uriagereka, 89–156. Cambridge, Mass.: MIT Press.

Chomsky, Noam. 2001. Derivation by phase. In *Ken Hale: A life in language*, ed. Michael J. Kenstowicz, 1–52. Cambridge, Mass.: MIT Press.

Chomsky, Noam, and Morris Halle. 1968. *The sound pattern of English*. New York: Evanston.

Collins, Chris. 1994. Economy of derivation and the generalized proper binding condition. *Linguistic Inquiry* 25:45–61.

Collins, Chris. 1996. *Local economy*. Cambridge, Mass.: MIT Press.

Drummond, Alex. 2010. Fragile syntax and sideward movement. Ms., University of Maryland.

Engelfriet, Joost. 1975. Bottom-up and top-down tree transformations — a comparison. *Mathematical Systems Theory* 9:198–231.

Fox, Danny. 1995. Economy and scope. *Natural Language Semantics* 3:283–341.

Fox, Danny. 2000. *Economy and semantic interpretation*. Cambridge, Mass.: MIT Press.

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

Gärtner, Hans-Martin. 2002. *Generalized transformations and beyond: Reflections on minimalist syntax*. Berlin: Akademie-Verlag.

Grodzinsky, Yosef, and Tanja Reinhart. 1993. The innateness of binding and coreference. *Linguistic Inquiry* 24:69–102.

Gécseg, Ferenc, and Magnus Steinby. 1984. *Tree automata*. Budapest: Academei Kaido.

Heim, Irene. 1998. Anaphora and semantic interpretation: A reinterpretation of Reinhart's approach. In *The interpretive tract*, ed. Uli Sauerland and O. Percus, volume 25 of *MIT Working Papers in Linguistics*, 205–246. Cambridge, Mass.: MIT Press.

Heim, Irene. 2009. Forks in the road to Rule I. In *Proceedings of NELS 38*, 339–358.

Hopcroft, John E., and Jeffrey D. Ullman. 1979. *Introduction to automata theory, languages, and computation*. Reading, Mass.: Addison Wesley.

Hornstein, Norbert. 2001. *Move! A minimalist theory of construal*. Oxford: Blackwell.

Huang, C.-T. James. 1982. Logical relations in Chinese and the theory of grammar. Doctoral Dissertation, MIT.

Johnson, David, and Shalom Lappin. 1999. *Local constraints vs. economy*. Stanford: CSLI.

Joshi, Aravind. 1985. Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In *Natural language parsing*, ed. David Dowty, Lauri Karttunen, and Arnold Zwicky, 206–250. Cambridge: Cambridge University Press.

Jäger, Gerhard. 2002. Gradient constraints in finite state OT: The unidirectional and the bidirectional case. In *More than words. A festschrift for Dieter Wunderlich*, ed. I. Kaufmann and B. Stiebels, 299–325. Berlin: Akademie Verlag.

Karttunen, Lauri. 1998. The proper treatment of optimality in computational phonology. Manuscript, Xerox Research Center Europe.

Kepser, Stephan, and Uwe Mönnich. 2006. Closure properties of linear context-free tree languages with an application to optimality theory. *Theoretical Computer Science* 354:82–97.

Keshet, Ezra. 2010. Situation economy. *Natural Language Semantics* 18:pp–pp.

Kobele, Gregory M. 2006. Generating copies: An investigation into structural identity in language and grammar. Doctoral Dissertation, UCLA.

Kobele, Gregory M. 2010. Without remnant movement, MGs are context-free. In *MOL 10/11*, ed. Christian Ebert, Gerhard Jäger, and Jens Michaelis, volume 6149 of *Lecture Notes in Computer Science*, 160–173.

Kobele, Gregory M., Christian Retoré, and Sylvain Salvati. 2007. An automata-theoretic approach to minimalism. In *Model Theoretic Syntax at 10*, ed. James Rogers and Stephan Kepser, 71–80. Workshop organized as part of the Europen Summer School on Logic, Language and Information, ESSLLI 2007, 6-17 August 2007, Dublin, Ireland.

Kolb, Hans-Peter, Jens Michaelis, Uwe Mönnich, and Frank Morawietz. 2003. An operational and denotational approach to non-context-freeness. *Theoretical Computer Science* 293:261–289.

Michaelis, Jens. 1998. Derivational minimalism is mildly context-sensitive. *Lecture Notes in Artificial Intelligence* 2014:179–198.

Michaelis, Jens. 2001. Transforming linear context-free rewriting systems into minimalist grammars. *Lecture Notes in Artificial Intelligence* 2099:228–244.

Müller, Gereon, and Wolfgang Sternefeld. 1996. A-bar chain formation and economy of derivation. *Linguistic Inquiry* 27:480–511.

Nakamura, Masanori. 1997. Object extraction in Bantu applicatives: Some implications for minimalism. *Linguistc Inquiry* 28:252–280.

Nunes, Jairo. 2004. *Linearization of chains and sideward movement*. Cambridge, Mass.: MIT Press.

Potts, Christopher. 2002. Comparative economy conditions in natural language syntax. Paper presented at the North American Summer School in Logic, Language, and Information 1, Workshop on Model-Theoretic Syntax, Stanford University (June 28), June 2002.

Potts, Christopher, and Geoffrey K. Pullum. 2002. Model theory and the content of OT constraints. *Phonology* 19:361–393.

Prince, Alan, and Paul Smolensky. 2004. *Optimality theory: Constraint interaction in generative grammar*. Oxford: Blackwell.

Reinhart, Tanya. 2006. *Interface strategies: Optimal and costly computations*. Cambridge, Mass.: MIT Press.

Rezac, Milan. 2007. Escaping the person case constraint: Reference-set computation in the ϕ-system. *Linguistic Variation Yearbook* 6:97–138.

Rizzi, Luigi. 1997. The fine-structure of the left periphery. In *Elements of grammar*, ed. Liliane Haegeman, 281–337. Dordrecht: Kluwer.

Rizzi, Luigi. 2004. Locality and left periphery. In *The cartography of syntactic structures*, ed. Adriana Belletti, volume 3, 223–251. New York: Oxford University Press.

Rogers, James. 1997. "Grammarless" phrase structure grammar. *Linguistics and Philosophy* 20:721–746.

Rogers, James. 1998. *A descriptive approach to language-theoretic complexity*. Stanford: CSLI.

Shieber, Stuart M. 2004. Synchronous grammars as tree transducers. In *TAG+7: Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms*, 88–95.

Shieber, Stuart M. 2006. Unifying synchronous tree adjoining grammars and tree transducers via bimorphisms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, 377–384.

Shieber, Stuart M., and Yves Schabes. 1990. Synchronous tree adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, 253–258.

Shima, Etsuro. 2000. A preference for move over merge. *Linguistic Inquiry* 375–385.

Stabler, Edward P., and Edward Keenan. 2003. Structural similarity. *Theoretical Computer Science* 293:345–363.

Sternefeld, Wolfgang. 1996. Comparing reference-sets. In *The role of economy principles in linguistic theory*, ed. Chris Wilder, Hans-Martin Gärtner, and Manfred Bierwisch, 81–114. Berlin: Akademie Verlag.

Szendrői, Kriszta. 2001. Focus and the syntax-phonology interface. Doctoral Dissertation, University College London.

Thomas, Wolfgang. 1997. Languages, automata and logic. In *Handbook of formal languages*, ed. Gregorz Rozenberg and Arto Salomaa, volume 3, 389–455. New York: Springer.

Toivonen, Ida. 2001. On the phrase-structure of non-projecting words. Doctoral Dissertation, Stanford, CA.

Wartena, Christian. 2000. A note on the complexity of optimality systems. In *Studies in optimality theory*, ed. Reinhard Blutner and Gerhard Jäger, 64–72. Potsdam, Germany: University of Potsdam.

Wilder, Chris, and Hans-Martin Gärtner. 1997. Introduction. In *The role of economy principles in linguistic theory*, ed. Chris Wilder, Hans-Martin Gärtner, and Manfred Bierwisch, 1–35. Berlin: Akademie Verlag.

Affiliation

Thomas Graf
Department of Linguistics
University of California, Los Angeles
tgraf@ucla.edu