Concealed Reference-Set Computation: How Syntax Escapes the Parser's Clutches

Running Head: Concealed Reference-Set Computation

Thomas Graf

Abstract: It has been conjectured that all properties of language beyond recursion can be motivated by interface requirements. One component in this setup is the parser, which is thought to give rise to a preference for computational parsimony. I discuss a mathematical result on reference-set computation, an (allegedly) non-parsimonious piece of machinery, that challenges this assumption and suggests that syntax can sometimes "trick" the parser in order to escape its demands. If reference-set constraints are construed as so-called tree transducers, they do not increase the power of the syntactic machinery. This in turn entails that they can be expressed purely in terms of the Minimalist feature calculus regulating Merge and Move, so it cannot be ruled out that syntax employs reference-set constraints, whereas the parser operates on their less demanding equivalents. In order to demonstrate the viability of this approach for the kind of reference-set constraints found in the literature, I give an implementation of Merge-over-Move.

Keywords: Transderivationality, Merge-over-Move, Minimalist grammars, tree transducers

Introduction

A core assumption of the biolinguistic program is that all properties of syntax beyond recursion are motivated by requirements imposed by other cognitive modules (Chomsky 2005; Hauser et al. 2002). This raises the question, though, whether these external factors merely restrict the shape syntax may take or, rather, uniquely determine it. In other words, are there multiple ways syntax may satisfy all interface requirements, or just one? The latter hypothesis is clearly the stronger one and thus more intriguing. As I will argue in this paper, though, it is untenable. In particular, I propose that syntax can trick the interfaces and avail itself of certain tools that conflict with their demands.

My argument is based on a case-study pertaining to the relation between the computability desiderata of the parser and the class of reference-set constraints (RCs; also known as trans-

derivational constraints or global economy conditions) RCs differ markedly from standard well-formedness conditions in that they depend on transderivational comparisons. Given a tree, they determine its set of competitors — called its *reference set* — and pick from said set the optimal tree(s) according to some economy metric. All other trees are filtered out. Among the RCs proposed in the literature one finds the Accord Maximization Principle (Schütze 1997), Avoid Structure (Rizzi 1997), Chain Uniformization (Nunes 2004), Focus Economy (Reinhart 2006), Merge-over-Move (Chomsky 2000), Pronouns as Last Resort (Hornstein 2001), Rule I (Reinhart 2006), Rule H (Fox 2000), Scope Economy (Fox 2000), the Shortest Derivation Principle (Chomsky 1995), Situation Economy (Keshet 2010), and several more. Nonetheless it has been argued that RCs are not part of narrow syntax due to the computational complexity of transderivational comparisons; an abundance of candidates needs to be computed, the majority of which is subsequently discarded (Collins 1996; Jacobson 1997; Johnson and Lappin 1999). It seems, then, that RCs are in conflict with the main requirement imposed by the parser, i.e. efficient computability, whence syntax must eschew them.

When studied from a mathematical perspective, however, the transderivational component of RCs turns out to be superfluous. Many RCs can be recast purely in terms of the Minimalist feature calculus, which does not increase the grammar's resource usage. This immediately implies that syntax could in principle flout the parser's demands by using RCs while the parser gets to operate with the more economical alternative. As a consequence, there are two variants of narrow syntax — with RCs and without them — that the interfaces have no reason to treat differently, contrary to the strong hypothesis formulated at the outset that the shape of syntax is uniquely determined by interface requirements.

My mathematical result hinges on the assumption that at least some RCs can be modelled by so-called linear bottom-up tree transducers. In order to convince the reader of the applicability of this formalism to linguistically motivated RCs, I give a rigorous implementation of Merge-over-Move (MOM; Chomsky 1995, 2000), a principle demanding that when given a choice between Merge and Move at some step in the derivation, Merge is to be preferred over Move unless this

would cause the derivation to crash later on. The strategy I follow in the implementation of MOM combines ideas that have been successfully used in Graf (2010a,b) to model two other RCs, the Shortest Derivation Principle (SDP; Chomsky 1995) and Focus Economy (Reinhart 2006). Most RCs in the syntactic literature can be viewed as a variation of one of these three constraints. The Accord Maximization Principle, for instance, is obtained from the SDP by prefering longer derivations over shorter ones. Avoid Structure (Rizzi 1997) resembles the SDP in that it also prefers simpler derivations but measures simplicity by the number of functional categories instead of the number of movement steps. These kinds of changes are innocent from a technical perspective, so it is safe to assume that linear bottom-up tree transducers do indeed provide a model for the majority of syntactic RCs.

The paper is laid out as follows: Sec. 1 provides a gentle introduction to the formal tools that form the basis of my argument, Minimalist grammars (MGs; 1.1) and linear tree transducers (1.2). I also describe the general strategy for modelling RCs, and why RCs can be recast without their transderivational component (1.3). After that, I work through a linguistically motivated example, MOM (2). The transducer for this constraint is described in intuitive terms (2.2), with the actual math relegated to the appendix. Interestingly, the most natural transducer implementation of MOM suggests a minor tweak to the constraint that has the welcome side-effect of doing away with certain undergeneration problems of MOM and its reliance on structured numerations, as is discussed in Sec. 2.3.

1 The Mathematical Argument

1.1 Minimalist Grammars

In an attempt to base my claims on as rigorous a foundation as possible, I use MGs (Stabler 1997, 2011) as a formal model of Minimalist syntax. Actually, this is a slightly inaccurate statement as there are many extensions of the MG formalism comprising a wide range of items from the syntactician's toolbox. Among them we find head movement, affix hopping, pied-piping,

covert movement, sideward movement, Agree, phases, superiority conditions, islands constraints, complexity filters and even Distributed Morphology. Surprisingly, though, all of them can be emulated by a bare-bones variant of MGs that only uses *Merge*, *Move* and a strengthened version of the *Shortest Move Constraint* (see Graf 2011, Kobele 2011, Stabler 2011 and references therein). Hence I will restrict my attention to these maximally simplified MGs.

Like the standard version of Minimalism predating the introduction of Agree in Chomsky (2000), MGs construct phrase structure trees from lexical items (LIs) via the operations Merge and Move, which in turn are triggered by features on those LIs. In contrast to early Minimalism, however, MGs are more explicit about the nature of features and the structure of LIs in general. First, there is a strict division between features that trigger Merge and features that trigger Move. Second, features come in two polarities, which I indicate by superscripted $+$ and $-$. Third, an LI may carry several instances of the same feature. Finally, the features of an LI must be checked in a prespecified order that may vary between LIs. A typical MG lexicon might look as follows.

men :: N^-	like :: D^+ D^+ V^-
the :: N^+ D^- nom^-	ε :: V^+ nom^+ T^-
what :: D^-	ε :: T^+ C^-
what :: D^- wh^-	do :: V^+ wh^+ C^-

The symbol before the double colon is the phonetic exponent of the LI; LIs marked with ε are phonetically null. After the double colon we find a string of features associated with the LI, with Merge features denoted by uppercase letters and Move features in lowercase letters. Negative Merge features correspond to category features, while positive Merge features are selector features. For instance, *the* selects a noun and is itself of category D. The verb *like* selects two DPs and is of category V. Crucially, *the* cannot be merged with *like* unless it selects a noun first because its first feature is N^-, not D^+. The category feature D^+ remains inaccessible until the selector feature N^- is deleted by Merge. Move features behave similarly: negative Move features (*licensee* features) mark the LI undergoing movement, positive Move features (*licensor* features) the target. As usual, movement of an LI displaces the entire phrase rather than just the head. Once again,

though, a feature cannot be operated on before all the features preceding it have been checked, irrespective of their polarity and whether they trigger Merge or Move. Given this basic feature calculus regulating Merge and Move, a tree is considered to be grammatical if all features have been successfully checked off except the category feature of the highest LI, which must be a C-head.

However, the number of configurations in which two features may enter a checking relation via Move is severely reduced by the Shortest Move Constraint (SMC): at no point in a derivation may two LIs have the same licensee feature as their first unchecked feature. The SMC entails that if some LI has an active movement licensor feature, there is exactly one LI with a matching licensee feature, otherwise the derivation crashes (there must be at least one matching feature, and the SMC rules out that there is more than one). So this constraint is essentially a strengthened version of Relativized Minimality (Rizzi 1990), with the side-effect that Move becomes deterministic. For instance, the SMC can be invoked to block superiority violations such as *What did you say who bought t*. Under the proviso that the feature string of both *who* and *what* is $-\mathrm{D}\ -\mathrm{wh}$, the derivation crashes right after *who* is merged with *bought what* due to an SMC violation, since both wh-words now have $-\mathrm{wh}$ as their first unchecked feature. Of course one has to ensure that *who* actually carries $-\mathrm{wh}$, and there are several constructions such as multiple wh-movement which seem to be incompatible with the SMC's strict interpretation of Relativized Minimality. Fortunately, though, these issues have been adressed in the literature (see Gärtner and Michaelis 2010 and Graf 2011, among others), so we need not worry about them here. The SMC, then, can be safely adopted as a means for introducing a notion of locality into MGs and keeping Move as simple as possible by making it deterministic.

A bare phrase structure produced by our example grammar that also involves wh-movement is given in Fig. 1, together with its corresponding derivation tree. The derivation tree is almost identical to the bare phrase structure tree except that (I) LIs are given with their feature component, (II) interior nodes are labeled by the name of the operation that applied at this stage in the derivation, and (III) instances of Move are indicated only by the label, i.e. the affected subtree remains in the

position where it was originally merged. The simplified representation is possible only because movement is deterministic in MGs, thanks to the SMC — at no point in the derivation may there be any ambiguity as to which constituent is targeted by an instance of Move. The reader is invited to explicitly verify that both the SMC and MGs' feature calculus are obeyed in the example tree.

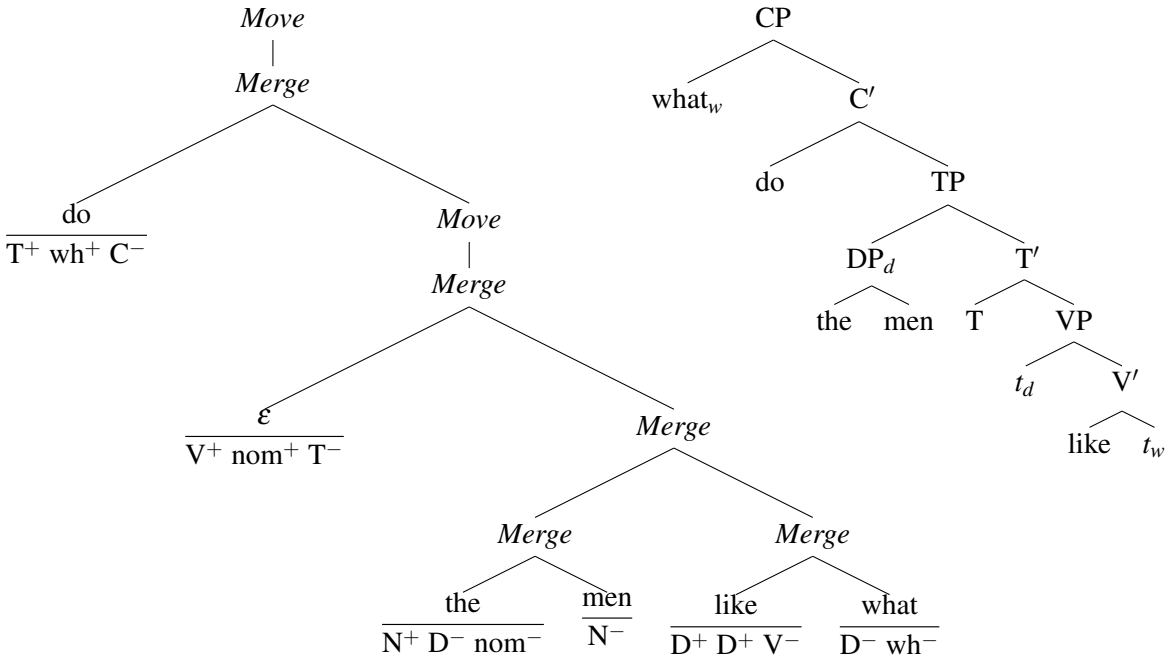

Figure 1: Bare phrase structures tree (right) and derivation tree (left) of a wh-question generated by the example MG

1.2 Linear Tree Transducers

While MGs serve as a formally rigorous model of Minimalist syntax, RCs are best understood in terms of linear bottom-up tree transducers (which I will henceforth refer to as "transducers" for the sake of brevity). Transducers can be viewed as a particular kind of SPE- or Aspects-style rewriting system. When handed a tree as its input, a transducer moves through said tree in a bottom-up fashion, from the leaves towards the root, possibly relabeling nodes, deleting subtrees, or inserting new structural material. Once it reaches the root, the end result of its modifications is either thrown away or returned as an output tree, depending on whether the end result is deemed well-formed. Sometimes there are several ways to manipulate an input tree, and in these cases the

transducer might create multiple output trees. The connection between transducers and rewriting systems brings about an intriguing shift in perspective regarding RCs: rather than filtering out suboptimal trees in a rather arcane, non-local way, RCs rewrite them into optimal ones using what may be considered a subclass of syntactic transformations.

Even though the main body of this paper is deliberately light in notation and the actual math confined to the appendix, transducers must be discussed in due detail to give the reader at least an intuitive grasp of their capabilities. Table 1 on the following page depicts the rules of a transducer for simple instances of wh-movement. Each consists of a left-hand side, a rewrite arrow, and a right-hand side. The left hand side varies depending on whether the transducer is at a leaf node or a non-terminal node. In the former case, it comprises only the label of said node, as in rules (1) and (2). Otherwise, it specifies the label of the current node, plus special symbols called *states* (by convention written as q plus some index) that immediately dominate the subtrees rooted by the daughters of the current node. The state symbols are not part of the input tree but were added by the transducer as a kind of temporary memory — when deciding which rewrite rule to apply, a transducer may only consider (I) the node it is currently at, and (II) the state symbols said node dominates. So rule (5), for instance, may be applied if and only if the current node is labeled TP, its left daughter is q_*, and its right daughter is q_{wh}.

The purpose of states is easier to fathom once one also takes the right-hand side and its interaction with the left-hand side into account. On the right-hand side of rules (3), (4) and (5), the states dominated in the left-hand side are gone. Instead, a new state was added on top of the new output subtree. Similarly, the right-hand sides of rules (1) and (2) each contain a state dominating the leaf node. This setup allows left-hand sides and right-hand sides to interact as follows in order to push states upwards through the tree during the rewrite steps: First, the transducers reads the currents node label and the states it dominates, if they exist. Depending on the applicable rewrite rules, it may leave this part of the tree unaltered (rule (1)), change the label (rule (2)), insert new structure (rule (5)), or delete a subtree (the last option is not featured in our example, but could easily be obtained from, say, rule (4) by removing one of the two subtrees in the right-hand side). Irrespec-

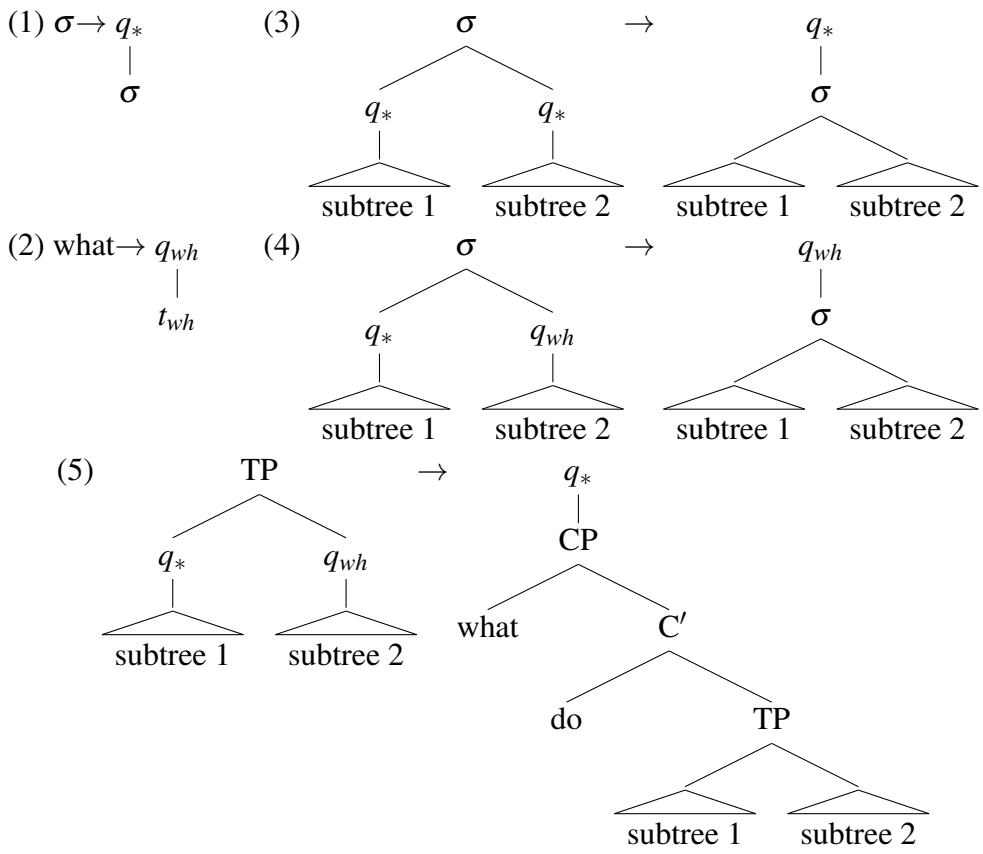

Table 1: Rules of a transducer for simplified wh-movement; only q_* is a final state

tive of how the subtree is rewritten, though, the transducer must put a state symbol on top of it, i.e. closer to the root. Note that this way of rewriting makes it necessary to traverse trees bottom-up. One starts by rewriting leaves adding states on top of them, then one rewrites the mothers of the leaves (which involves removing the old states and again adding a new one on top), after that the next higher mothers, and so on, until one finally reaches the root. The transducer deems the output tree well-formed only if the state dominating the root node is a *final state*. For each transducer one has to specify in advance which states are final states.

Let us work through the example in Tab. 1 in greater detail now. As I mentioned in passing before, the transducer is supposed to model very simple instances of wh-movement, similar to the MG we encountered in the previous section (wh-movement was chosen because it should be sufficiently familiar to all readers that they can fully focus their attention on the mechanics of the transducer). Only two states are used, q_* and q_{wh}. The former reminds the transducer that it did

not change anything in the subtree dominated by q_* — we may call it the identity state — whereas q_{wh} signals that somewhere in the subtree it dominates, a wh-word was replaced by a trace.

The five rules of the transducer can now be paraphrased as follows. First, note that σ is used as a shorthand for any label, so an instruction to rewrite σ as σ instructs the transducer to keep the current label. Consequently, rule (1) tells the transducer that if it is at a leaf node, it should leave said node unaltered and record this decision by adding the state q_* on top. Rule (2), on the other hand, allows for leaf nodes labeled *what* to be rewritten by wh-traces. Naturally we add the state q_{wh} this time. Crucially, the two rules are not in conflict, and the transducer may choose freely between rule (1) and (2) whenever it encounters a leaf labeled *what* (since σ matches any label). Hence the transducer creates several output trees for inputs with wh-words, some with wh-in-situ and some with wh-movement. Rule (3) and (4) are fairly unremarkable insofar as they merely ensure that the transducer does not manipulate non-terminal nodes and that q_{wh} is percolated upwards as necessary. If we did not take care to carry along q_{wh} at every step of the rewriting process, then the transducer would "forget" that it had replaced a wh-word by a trace earlier on. That is to say, it would merely remove wh-words rather than displace them. Finally, rule (5) tells the transducer to add a CP with the wh-word on top of a TP if rule (2) was applied at some earlier point. Note that if q_{wh} is a final state, rule (5) need never apply since output trees in which the transducer failed to switch back from q_{wh} into q_* before reaching the root node would also be considered acceptable. Hence only q_* may be a final state if we want wh-words to be reinserted into the tree after they have been replaced by traces.

A transduction using all five rules is given in Fig. 2 on the following page. Except for deletion, it shows off all the capabilities of transducers that will be needed for RCs, in particular relabelings, the insertion of new material, and the ability to use states to both memorize limited amounts of structural information and decide whether output trees should be accepted or discarded.

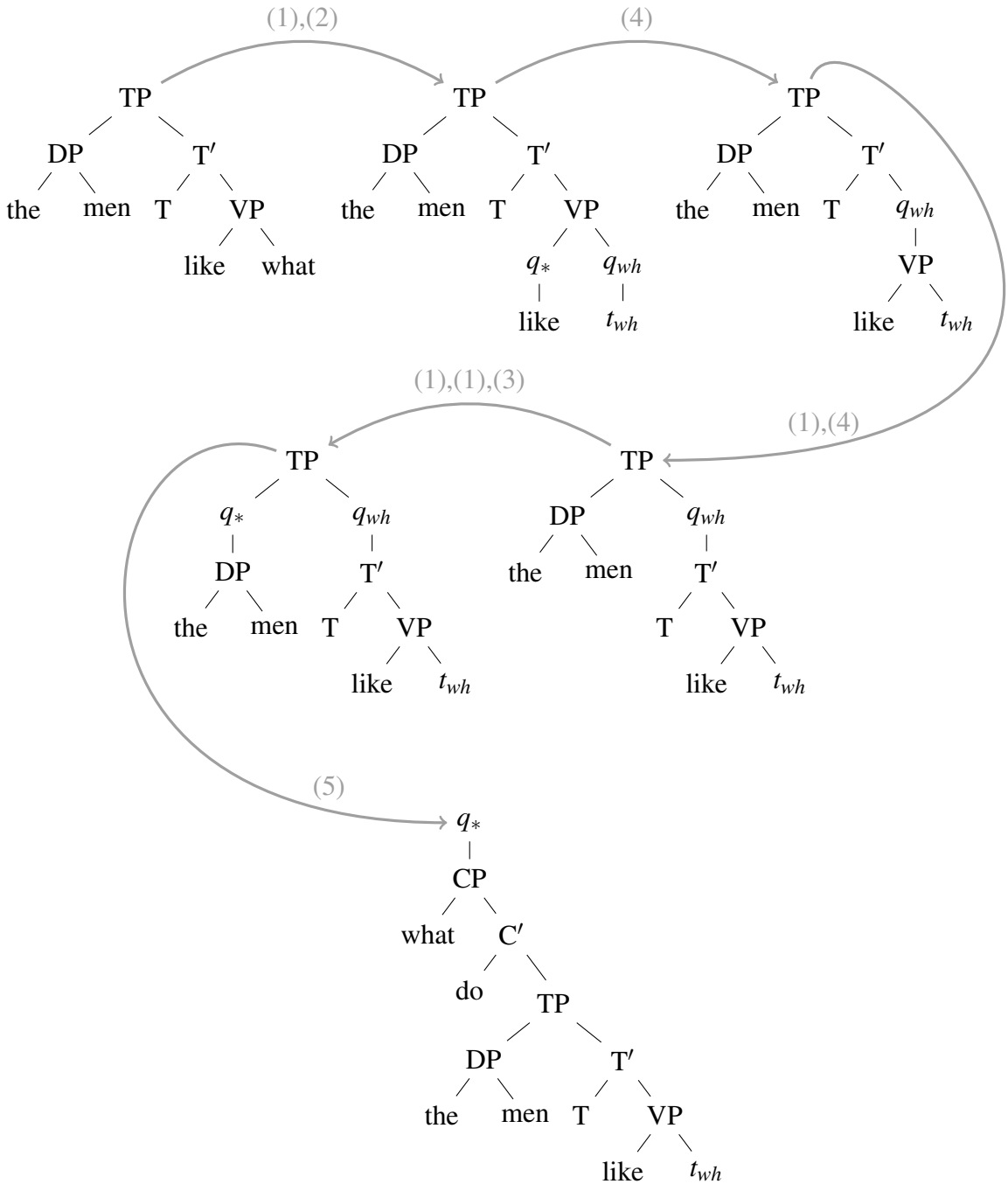

Figure 2: Example of transduction for simple wh-movement

1.3 Putting it All Together

So far we have seen MGs as a formalization of Minimalist syntax and transducers as a potential mathematical model of RCs, but it is still unclear how the two combine to reveal something fundamental about RCs. The answer, albeit building on highly technical insights (Engelfriet 1975; Graf 2011; Kobele 2011), is simple: if a transducer is used like an RC to rewrite suboptimal derivations of a given MG G into optimal ones, then the set of these optimal derivations is exactly the set of well-formed derivations of some other MG G'. This implies that RCs that can be modelled by transducers do not increase the power of the MG formalism, and consequently they can be expressed purely in terms of the feature calculus driving Merge and Move.

It still needs to be shown, though, that RCs can be implemented as transducers. To this end, I produce a transducer for MOM in the next section. The procedure I use exploits a fact about transducers proved by Engelfriet (1975). Given two linear transducers α and β, we can concatenate them into a single linear transducer τ such that for I and O the sets of inputs and outputs of α, respectively, applying τ to I yields the same set as applying β to O. The reverse is also true: every transducer τ that rewrites a set I of trees as set O can be split into two transducers α and β such that applying α to I yields a set Z, which in turn is rewritten as O by β. So transducers can be concatenated or spliced arbitrarily often and in any conceivable way, the end result of the rewriting procedure will always be the same. This is helpful for our purposes because it allows for RCs to be defined in a piece-wise manner, breaking them into small, easily understood parts that can then be recombined into one big transducer representing the constraint as a whole.

This has already been accomplished for Focus Economy and the Shortest Derivation Principle (Graf 2010a,b), which constitute special cases of the following model. In order to capture an RC, one defines four transducers that are meant to apply serially one after the other. The first one maps each tree to an underspecified structure that implicitly encodes the commonalities of all trees belonging to the same reference set. The second one turns these underspecified structures into fully specified derivation trees again. So when these two transducers are executed one after another, each tree in the input language is rewritten as the trees in its reference set. The third transducer then

implements the economy metric by rewriting suboptimal trees as optimal ones. So if the first three transducers are run in series, every tree in the input language is rewritten as an optimal tree (if there are several, all of them will be generated). Finally, the fourth transducer discards all trees that weren't already included in the input language to prevent accidental overgeneration. In the remainder of this paper, I demonstrate that a strategy along these lines also works for MOM, although the details differ notably from those for Focus Economy and the Shortest Derivation Principle.

2 A Practical Example: Implementing Merge-over-Move

2.1 Merge-over-Move Explained

Modelling MOM is complicated by the fact that there are multiple versions of the constraint, which are seldom carefully teased apart in the literature. Before we can discuss the transducer implementation (Sec. 2.2) and evaluate its faithfulness and empirical adequacy (Sec. 2.3), then, some variant must first be agreed upon as the measuring rod.

All definitions of MOM seek to formalize the following intuition: 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.

(1) 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 the Minimalist framework of Chomsky (1995), we start out with a multiset of LIs — 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, (1a) and (1c) are easy to derive. Consider (1c) first. It starts out with the numeration {seems, to, be, a, man, in, the, garden} (ignoring phonetically null functional categories). Multiple applications

of Merge yield the phrase [$_{TP}$ 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 [$_{TP}$ a man to be t_{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.

Things are slightly more involved for (1a). 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 [$_{TP}$ to be [$_{DP}$ a man in the garden]]. Now there are two possible continuations: (I) Merger of *there*, which is later followed by moving *there* into the specifier of *seems* and thus yields the grammatical (1a), or (II) 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 (1b). MOM rectifies this overgeneration problem by barring movement of *a man* into the specifier of *to be* due to the existence of a more economical option, namely merging *there*. At the same time, MOM does not block (1c) because it involves no choice between Merge and Move.

Depending on the conditions under which MOM is thought to apply, two variants can be distinguished. The *indiscriminate* version (iMOM) always prefers Merge over Move. As a result, it picks the most economical derivation, even if said derivation will eventually crash. The *cautious* version (cMOM), on the other hand, follows the original proposal of Chomsky (1995) in that only convergent derivations can be chosen. Both iMOM and cMOM use the *Identity of Numerations Condition* (INC) for determining a derivation's *reference set*, i.e. the set of derivations it competes with. The INC states that the reference set of a derivation d contains all the derivations that can be built from the same numeration as d, and only those.

Evidently cMOM has a distinctly global flavor to it: if MOM is evaluated at every step of the derivation, i.e. before the derivation is completed, then the only way to predict which competing derivations will crash later on — and thus must be discarded for the comparison — is via a filter based on unlimited look-ahead or reference-set computation. The iMOM variant, by contrast,

provides a strictly local alternative. It contains not even a modicum of global economy, as it simply prefers Merge over Mover whenever there is a locally licensed choice between the two, no matter what the eventual outcome.

For simple cases like (1) both MOM variants make the right predictions. However, there are cases where they behave differently (Shima 2000).

(2) 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. Subsequently, a decision has to be made as to whether one wants to move *John* into SpecTP or base-merge the expletive instead. The iMOM variant picks the base-merger route, so we end up with [it [is [how likely John to win]]. This means that iMOM will always block the grammatical (2a), irrespective of how the derivation continues.

The behavior of cMOM is more refined. Assume that base-merger of *it* is favored, yielding [it [is [how likely John to win]]. After this phrase is merged with *asked* and *is*, *John* must move into the specifier of the matrix TP to get its case feature checked. This is when the implicit look-ahead of cMOM comes into play. Unless moving *John* is barred for independent reasons, the derivation for (2b) won't crash, so the initial choice of merging *it* rather than moving *John* won't be discarded by cMOM. As a result, (2b) incorrectly blocks (2a). However, if (2b) is illicit for independent reasons — say, because moving *John* out of [how likely John to win] constitutes an island violation — then cMOM won't consider base-merger of *it* a viable alternative to Move and only (2a) is generated.

There are also cases where both variants make the wrong predictions (Wilder and Gärtner 1997).

(3) 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 sentences are assembled from the same numeration, so (3b) should block (3a), since the for-

mer converges and violates MOM at a later derivational step than the latter. In order to account for such cases, Chomsky (2000) stratifies numerations such that each CP has its own subnumeration. In the case at hand, (3a) is built from the numeration {{there, was, a, rumor, in, the, air}, {that, was, a, man, in, the, room}}, and (3b) 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 generated from the latter.

So now another parameter of MOM must be taken into account. A *restricted* version of MOM (rMOM) is part of a grammar where every CP has its own numeration. An *unbounded* version (uMOM) belongs to a grammar with one big, unstructured numeration for the entire sentence. In this system, cuMOM is the version of MOM introduced in Chomsky (1995), iuMOM is its local counterpart, and crMOM is the modification put forward in Chomsky (2000). Out of these three alternatives, crMOM is arguably superior with respect to empirical coverage, whence it will be the starting point for the transducer implementation.

2.2 A Transducer Model of MOM

The general strategy for modelling MOM is slightly altered from the schema I outlined in Sec. 1.3. Just as described there, I first give a transducer that rewrites derivations as a certain kind of underspecified tree. The idea is that the latter provides an abstract encoding of the former's reference set. These underspecified trees are then immediately turned into optimal derivations (in contrast to the original schema, which posits some intermediate steps). By mapping all derivations to underspecified trees from which only optimal derivations can be obtained, every derivation that is suboptimal with respect to MOM is filtered out — or more precisely, rewritten as an optimal one.

Every variant of MOM, though, poses one major difficulty for transducers, namely the defintion of reference sets via the INC. Consider the utterance *John loves Mary*. Depending on how we interpret the notion of "identity of numerations" invoked by the INC, *John loves Mary* might have the same numeration as *Mary loves John*, so they should be in the same reference set. But

the underspecification format I propose cannot encode both derivations in a single tree. In fact, no transducer can map a given tree to the counterpart in which subject and object have been switched unless there is an upper bound on their respective size; this is called the inability to perform *local rotations*. Since DPs can be nested and may contain CPs, natural language arguments have unbounded size, so if the INC is an indispensable part of MOM, one would expect local rotations to be unavoidable, too. This is not the case.

Consider (1) again. MOM's objective is to explain why (1b) is ungrammatical, and it does so by using a metric that makes it lose out against (1a). The grammaticality of (1c), on the other hand, follows from the fact that its numeration differs from that of (1a) and (1b), whence it also belongs to a different reference set by the INC. But identity of numerations is a rather indirect encoding of the relationship that holds between (1a) and (1b). A simpler and more intuitive condition emerges when we look at their derivation trees (see Fig. 3). *Modulo* the feature specifications of the LIs, the only difference between the derivation trees of (1a) and (1b) is the timing of Move and *there*-Merger. Switching the positions of these two operations in the derivation does not constitute an instance of local rotation because the size of the moved subtree is at most 2 (*Merge* plus its left daughter *there*) and hence finitely bounded.

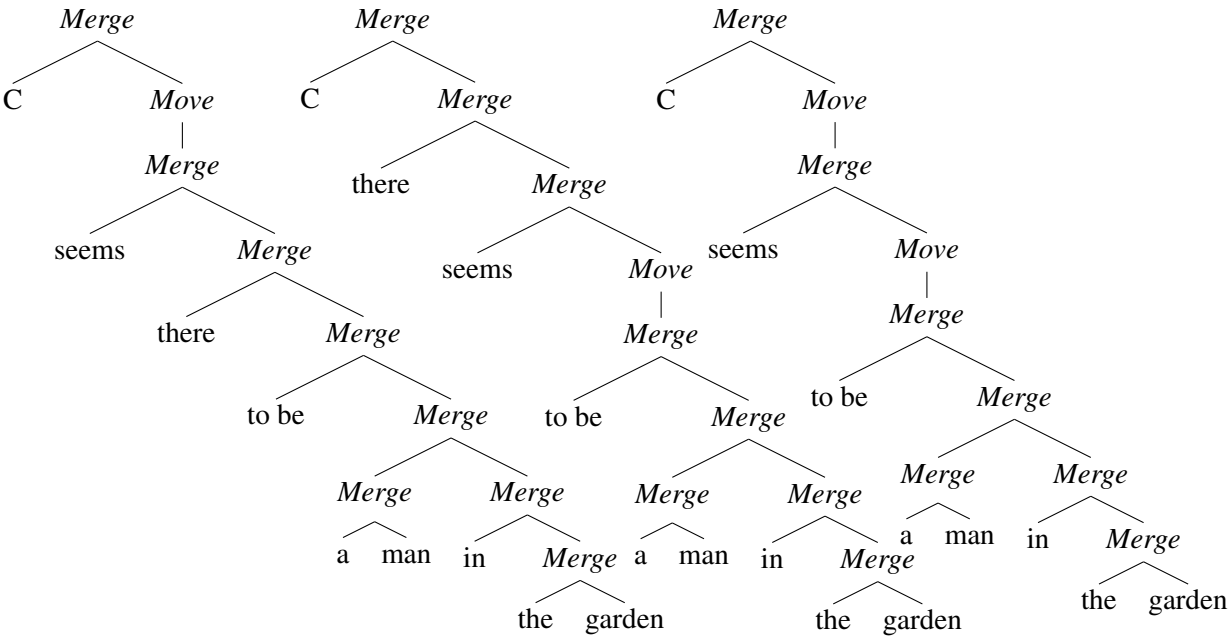

Figure 3: Derivation trees of (1a), (1b) and (1c); features are omitted

Recall that we want the transducer to eventually rewrite the suboptimal (1b) as the optimal (1a). As we just saw, this does not involve local rotations, contrary to what one would expect if the INC was the weakest working solution. But if the INC is indeed ill-suited to defining reference sets, what should supplant it? There are in fact infinitely many alternatives that all yield the same behavior. In my opinion, though, the most elegant solution is to give up the original partition of reference sets and make (1c) a competitor of (1a) and (1b).

Compare the derivations of (1a) and (1b) to that of (1c). The former are essentially the result of non-deterministically replacing one instance of Move in the derivation tree of (1c) by merger with expletive *there*. Strikingly, though, rewriting the lower occurrence of Move yields the grammatical (1a), whereas rewriting the structurally higher occurrence gives rise to the ungrammatical (1b). Suppose then, that we have an underspecified representation of all three derivations which is identical to (1c) except that the two instances of *Move* are replaced by a new symbol □. Now if we have a transducer that may rewrite □ as *Move* or merger of *there*, then all three derivations can be obtained from the underspecified tree. However, if we furthermore require that □ must be rewritten as *Move* if some other □ has already been rewritten as *Move* earlier on, then only (1a) and (1c) are possible outputs. The derivation for (1b) will never be generated — speaking in terms of RCs, it is filtered out as suboptimal with respect to MOM. From a linguistic perspective, MOM has been reduced to a local well-formedness condition on derivation trees: *Move* must not derivationally dominate an instance of *there*-merger.

The idea just outlined is captured as follows (see the appendix for a more rigorous description): First, our input language is the set of derivation trees of some MG. Then we use a transducer α to map each input derivation to some underspecified tree. For the sake of clarity, α is decomposed into two smaller transducers, *Remove Features* and *Underspecify*.

Remove Features strips away all features from the LIs except their category features. This step is indispensable because every instance of Merge or Move is tied to specific features in an MG, so any two derivations that differ with respect to the timing of Merge and Move also differ in the feature make-up of their LIs. Unless all those features are removed, derivations that should compete

with respect to MOM won't be mapped to the same underspecified tree. Category features are kept to ensure that two non-competing derivations with homophonic LIs have distinct underspecified representations. Note that since *Remove Features* involves only relabeling of leaf nodes, it can easily be carried out by a transducer.

Underspecify has slightly more work to do, but only inside TP, where it deletes expletive *there* and rewrites the corresponding Merge node as $\square$. Move nodes in this domain are also rewritten as $\square$. States must be used to detect which nodes must be rewritten or deleted. If a leaf's label is an LI of category T, C or simply *there* the transducer adds the state q_t, q_c, or q_{there} on top of it, respectively. Otherwise it inserts the neutral state q_*. The state q_t is percolated until a sister with state q_c is encountered, which indicates the beginning of the CP. The transducer has several rules for interior nodes, but only two of them do any actual work. The first one rewrites Move-nodes dominating the state q_t as $\square$, i.e. TP-internal Move nodes are relabeled $\square$. The second one applies to Merge-nodes dominating states q_t and q_{there}. This configuration indicates a TP-internal Merge-site of expletive *there*, so we delete the subtree deleted by q_{there} and change the label from *Merge* to $\square$. As *Underspecify* involves only relabelings and deletion of subtrees, it too is a transducer as defined in Sec. 1.2.

The underspecified trees created by α are then turned into fully specified ones again by the transducer β. As we saw in our discussion of (1a)–(1c), β only recovers the optimal derivations and thus enforces MOM. Just as with α, the workload of β is distributed over two smaller transducers, *Path Condition* and *Restore Features*.

Path Condition is the essential step that captures MOM's economy metric. Upon encountering a $\square$, it may rewrite it as *Move* or Merger of *there*, but with the added condition that once a $\square$ node has been replaced by *Move*, all remaining instance of $\square$ in the same TP have to be rewritten as *Move*, too. Formally, this is handled by having the transducer switch between three different states: q_*, q_{move}, and q_c. The first one is the default and merely encodes that no $\square$ has been rewritten as a Move node yet. For any $\square$ dominating q_*, the transducer may freely choose between *Move* or Merger of *there*. Once it opts for the former, though, it must project the state q_{move}, and

every □ dominating this state must be rewritten as *Move*. The only way for the transducer to stop percolating q_{move} and switch back into q_* is for it to reach a CP, which is encoded by the state q_c. This dependency on q_c "resets the transducer" for each CP and thus avoids overapplication of MOM, similar to the subnumerations of Chomsky (2000).

Two example runs of *Path Condition* are given in Fig. 4 and 5. Note how the transducer is locked into state q_{move} in Fig. 5 after rewriting the lower instance of □ as *Move* and consequently must also rewrite the higher instance as *Move*, making it impossible to generate the derivation tree for the ungrammatical *There seems a man to be in the garden*. These examples also illustrate that only relabelings and insertions of finitely-bounded subtrees are employed, which — as we saw with our toy transducer for wh-movement — are no problem for a transducer.

The last element of our chain of transducers is *Restore Features*, which undoes the effects of *Remove Features*, unsurprisingly. This is a non-deterministic process, though, and thus may overgenerate. For instance, the MG on page 4 allows for *what* to be reinstantiated with or without a wh^- feature. Hence *Restore Features* generates many ill-formed derivation trees that have to be filtered out later on. The required mechanism also takes care of another source of overgeneration. The non-deterministic rewriting of □ by *Path Condition* allows for two occurrences of □ to be rewritten as *there*, which yields the (derivation tree of the) ungrammatical *there seems there to be a man in the garden*. Both kinds of overgeneration are prevented by discarding all derivation trees that aren't licensed by the original grammar. Formally, this is handled by intersecting the output language of β with the input language of α, which can also be done by a transducer.

2.3 Empirical Evaluation

As I will show now, the transducer conception of MOM (tMOM) not only performs just as well as crMOM (Chomsky 2000), it is also less prone to undergeneration. The examples in Fig. 4 and 5 already demonstrate that tMOM successfully accounts for simple expletive/non-expletive alternations as in (1). So let us look at the more complex scenario in (3), repeated here as (4), which prompted the introduction of stratified numerations in the first place.

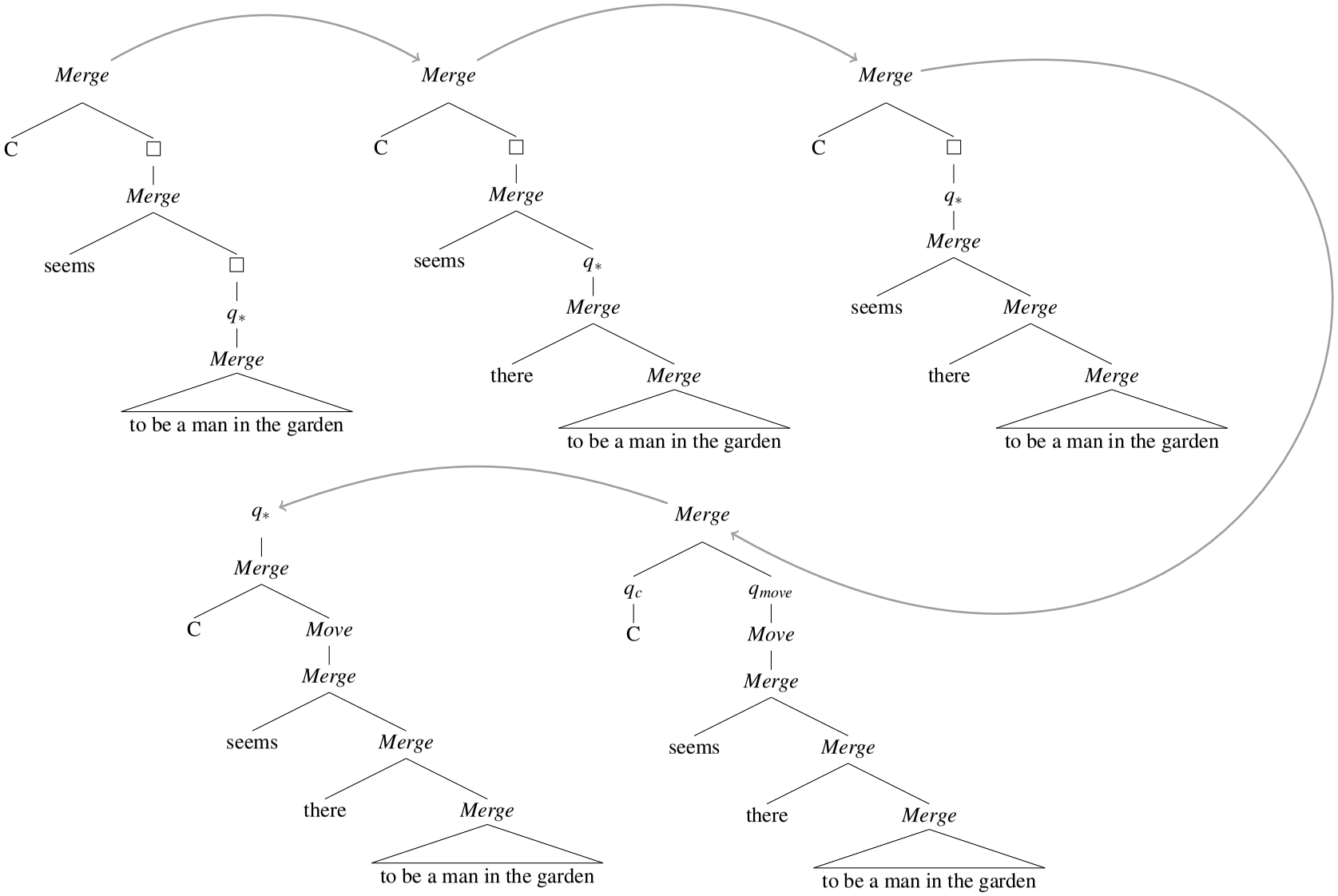

Figure 4: Rewriting the underspecified derivation tree into the derivation tree of *There seems to be a man in the garden*

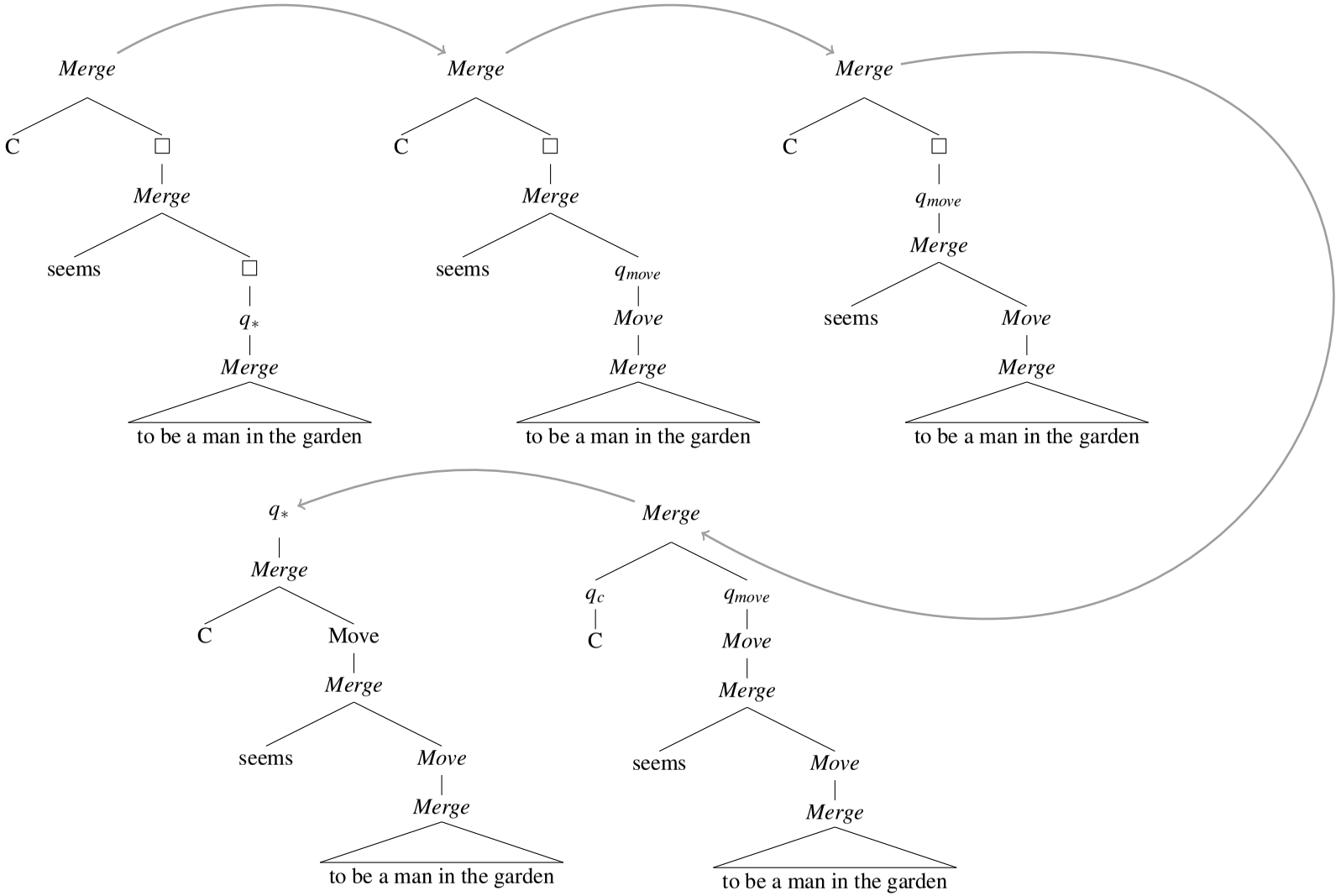

Figure 5: Rewriting the underspecified derivation tree into the derivation tree of *A man seems to be in the garden*

(4) 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. cuMOM and iuMOM); in the absence of stratified numerations the INC puts (4a) and (4b) in the same reference set, and as a result (4a) blocks (4b).

Under tMOM, on the other hand, (4) is a straightforward generalization of the pattern in (1). The underspecified tree of both sentences is shown in Fig.6. When it is expanded to full derivations again, all four logical possibilities are available: (I) *there*-insertion in both CPs, (II) Move in both CPs, (III) *there*-insertion in the lower CP and Move in the higher one, and (IV) 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 □ as Move"-state q_O after rewriting the label □ as *Move*, switches back into the neutral state q_* after having passed through the CP headed by *that*. Thus when it encounters the second □ 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 original MG, the following sentences are generated, all of which are grammatical:

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

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.

It is worth mentioning that tMOM differs from all other variants in how it groups these sentences into reference sets. The INC of the unstratified cuMOM and iuMOM entails that these four sentences belong to three distinct equivalence classes, one containing (5a), one containing (5b) and (5c), and one containing (5d). The crMOM variant with its 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 most intuitive route to take.

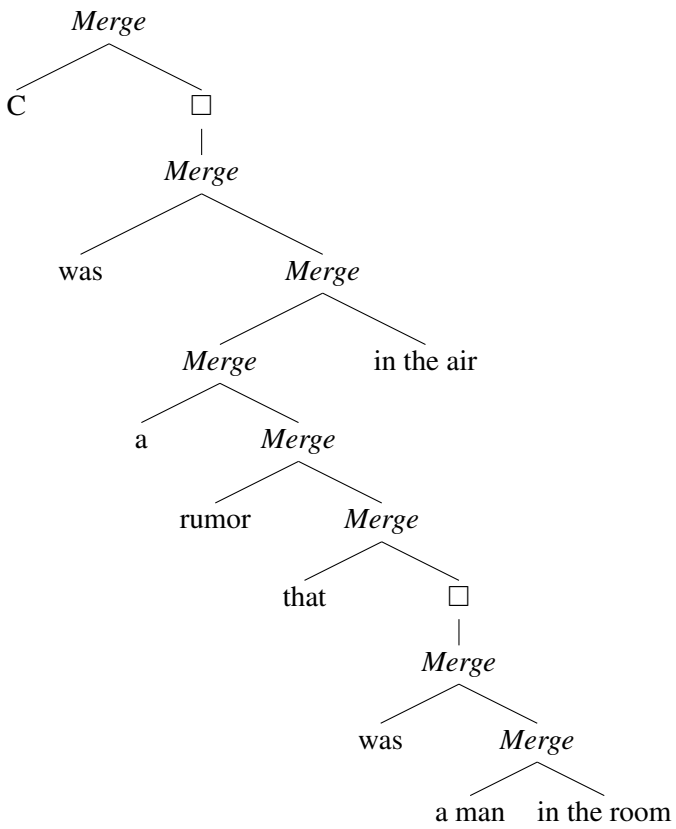

Figure 6: Underspecified Derivation Tree of (4a) and (4b)

Moreover, tMOM avoids certain undergeneration problems plaguing all other MOM variants, including crMOM (Shima 2000).

(6) 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.

Recall from our discussion in the previous section that iMOM always discards (6a), whereas c(r)MOM only does so if the derivation for (6b) does not crash (and the two even have identical stratified numerations). But tMOM never blocks it, because the grammaticality of the Merge variant has no bearing on the availability of the Move variant. This is so because tMOM is less of an economy-based dispreference for Move 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. The sequence of Merge and Move nodes in (6a) forms a licit pattern — Move is never followed by Merge inside a TP-domain — and thus tMOM will never filter out this derivation. It is striking that even though

this deviation from crMOM was prompted by the desire to keep tMOM as simple as possible (a completely faithful implementation of crMOM is possible, but would require parts of the Shortest Derivation Principle transducer of Graf 2010b), it actually is better behaved empirically.

Despite this small advantage, though, the behavior of tMOM still closely resembles that of crMOM. In particular, tMOM also fails to mark (6b) as ungrammatical. So both cMOM and tMOM require that the initial MG contains some locality condition that rules out (2b) in order to make the correct predictions. A natural candidate would be the islandhood of [how likely John to win].

Further assumptions about the input grammar must also be made to rule out cases of superraising like (7a) and multiple occurrences of *there* as in (7b).

(7) a. * A man seems there to be in the room.

b. * There seems there to be a man in the room.

On a conceptual level, this is a defensible move as the deviancy of these examples does not seem to be directly related to MOM. However, if we really wanted to incorporate such restrictions into MOM, at least the ban against double *there* can easily be accommodated by changing from a "once you opt for *Move*, you have to stick with it" version of *Path Condition* to the stricter "once you have rewritten a $\square$, all higher occurrences of $\square$ in this CP must be rewritten as *Move*" (this is accomplished by replacing the rule $\square(q_*(x)) \to q_*(Merge(\text{there}, x))$ in the appendix by the minimally different $\square(q_*(x)) \to q_o(Merge(\text{there}, x))$).

2.4 Technical Summary

RCs can be implemented using a tree rewriting formalism called (linear bottom-up tree) transducers. Transducers may relabel nodes, delete subtrees, or insert new subtrees of bounded size. They do so by traversing the tree from the leaves towards the root, inserting special state symbols at each step that serve as a storage for a limited amount of information about the subtree they dominate. Modelling RCs as transducers means that suboptimal derivations aren't filtered out but

rather turned into optimal ones. In the case of MOM, this is handled by a cascade of transducers that first turns derivations into an underspecified format omitting, in the TP domain, Move nodes and merger of *there*. Crucially, derivations that are competing with respect to MOM all share the same underspecified representation. After that, underspecified representations are turned into fully specified derivations again, but the transducer carrying out this rewrite step is constrained in such a way that only optimal derivations can be obtained this way. More precisely, no TP may contain a Move node that is dominated by a *there*-Merge node in the same TP. This implementation of MOM has several empirical advantages over the versions proposed in the literature. More importantly, though, transducers do not increase the power of Minimalist syntax as formalized in Stabler (1997, 2011). Consequently, MOM and other RCs can be reimplemented using only Merge, Move, and the feature calculus, so there is an equivalent but computationally more parsimonious way of ruling out the illicit structures.

Conclusion and Further Discussion

This paper started with a very general question — how much influence do the interfaces have on the shape of narrow syntax — and used RCs to debunk the strongest possible hypothesis, namely that syntax is uniquely determined by interface requirements. The foundation of the argument is a mathematical theorem that at least some RCs can be emulated by the Minimalist feature calculus combined with Merge and Move, which contradicts the assumption in the literature that RCs are computationally costly, whence they should be in violation of interface requirements. In order to demonstrate that this subclass of RCs contains constraints from the syntactic literature, I showed how MOM can be implemented using the relevant formal tools (Minimalist grammars and linear tree transducers). In sum, there are syntactic RCs that can be recast with less demanding machinery, so it cannot be ruled out that said RC is used by syntax while the component that imposes the stringent computability demands gets to operate with the local correspondent. It is conceivable, then, that syntax tricks the interfaces whenever it can conceal its offenses against their requirements.

References

Chomsky, Noam. 1995. *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. 2005. Three factors in language design. *Linguistic Inquiry* 36:1–22.

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

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

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

Gärtner, Hans-Martin, and Jens Michaelis. 2010. On the treatment of multiple-wh-interrogatives in minimalist grammars. In *Language and logos*, ed. Thomas Hanneforth and Gisbert Fanselow, 339–366. Berlin: Akademie Verlag.

Graf, Thomas. 2010a. Reference-set constraints as linear tree transductions via controlled optimality systems. In *Proceedings of the 15th Conference on Formal Grammar*. To appear.

Graf, Thomas. 2010b. A tree transducer model of reference-set computation. *UCLA Working Papers in Linguistics* 15:1–53.

Graf, Thomas. 2011. Closure properties of minimalist derivation tree languages. In *LACL 2011*, ed. Sylvain Pogodalla and Jean-Philippe Prost, volume 6736 of *Lecture Notes in Artificial Intelligence*, 96–111.

Hauser, Marc, Noam Chomsky, and W. Tecumseh Fitch. 2002. The faculty of language: What is it, who has it, and how did it evolve? *Science* 298:1569–1579.

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

Jacobson, Pauline. 1997. Where (if anywhere) is transderivationality located? In *The limits of syntax*, ed. Brian D. Joseph, Carl Pollard, Peter Culicover, and Louise McNally, 303–336. Burlington, MA: Academic Press.

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

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

Kobele, Gregory M. 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. In *LACL 2011*, ed. Sylvain Pogodalla and Jean-Philippe Prost, volume 6736 of *Lecture Notes in Artificial Intelligence*, 129–144.

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

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

Rizzi, Luigi. 1990. *Relativized minimality*. Cambridge, Mass.: MIT Press.

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

Schütze, Carson. 1997. INFL in child and adult language: Agreement, case and licensing. Doctoral Dissertation, MIT.

Shima, Etsuro. 2000. A preference for Move over Merge. *Linguistic Inquiry* 375–385.

Stabler, Edward P. 1997. Derivational minimalism. In *Logical aspects of computational linguistics*, ed. Christian Retoré, volume 1328 of *Lecture Notes in Computer Science*, 68–95. Berlin: Springer.

Stabler, Edward P. 2011. Computational perspectives on minimalism. In *Oxford handbook of linguistic minimalism*, ed. Cedric Boeckx, 617–643. Oxford: Oxford University Press.

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.

Appendix

I adopt the definition of MGs and Minimalist derivation tree languages (MDTLs) given in Graf (2011). As shown there and in (Kobele 2011), MDTLs are (almost) closed under linear tree

transductions, which entails that the image of an MDTL under a linear tree transduction is itself an MDTL (given a suitable refinement of the feature calculus).

The standard definition of linear bottom-up tree transducers is given below. The reader is referred to Engelfriet (1975) for further details, in particular for the rather cumbersome definition of the tree translations realized by a given transducer.

Definition 1. A *linear (bottom-up) tree transducer (lbutt)* is a 5-tuple $A := \langle \Sigma, \Omega, Q, Q', \Delta \rangle$, 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 $f(q_1(x_1), \ldots, q_n(x_n)) \to q(t)$, where $f \in \Sigma$ is a symbol of rank n, $q_1, \ldots, q_n, q \in Q$, t is a tree with labels drawn from $\Omega \cup \{x_1, \ldots, x_n\}$, and each x_i occurs at most once in t.

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 $\ldots$ or $\alpha_z(\beta_{z'}, \ldots, \zeta_{z''})$".

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

$$\sigma(q_i(x), q_j(y)) \to q_a(\sigma(x,y)) \qquad \sigma(q_i(x), q_j(y)) \to q_c(\sigma(x,y))$$
$$\sigma(q_j(x), q_k(y)) \to q_a(\sigma(x,y)) \qquad \sigma(q_j(x), q_k(y)) \to q_c(\sigma(x,y))$$
$$\sigma(q_a(x), q_c(y)) \to q_a(\sigma(x,y)) \qquad \sigma(q_a(x), q_c(y)) \to q_c(\sigma(x,y))$$
$$\sigma(q_b(x), q_c(y)) \to q_a(\sigma(x,y)) \qquad \sigma(q_b(x), q_c(y)) \to q_c(\sigma(x,y))$$

I now turn to the implementation of MOM by decomposing it into four linear transducers. I assume that the input to MOM is given by the MDTL L of some MG G with lexicon *Lex*. The first transduction to apply is *Remove Features*. As this transducer only relabels leaf nodes, I omit a full definition.

Definition 2. *Remove Features* is the deterministic (one-state) transducer that maps each LI $l := \sigma :: \gamma c \delta \in Lex$ to $l' := \sigma_c$, where c is a category feature and both γ and δ are strings of features. The set of these simplified LIs is denoted by Λ.

Even though the definition of an MG allows for an LI to have several category features or none at all, no such LI can ever appear in a well-formed derivation tree (cf. Graf 2011; Kobele 2011). Hence *Remove Features* is well-defined. If for some reason multiple category features (or their absence) are indispensable, the transduction can be extended such that each LI is subscripted by the n-tuple of its n category features. In either case, the map defined by *Remove Features* is many-to-one, so Λ is finite by virtue of the finiteness of *Lex*.

Definition 3. *Underspecify* is the lbutt $\mathscr{U}$, where $\Sigma_{\mathscr{U}} := \Lambda \cup \{Merge, Move\}$, $\Omega_{\mathscr{U}} := \Sigma_{\mathscr{U}} \cup \{\Box\}$, $Q := \{q_*, q_c, q_t, q_{there}\}$, $Q' := \{q_*\}$, and $\Delta_{\mathscr{U}}$ consists of the rules below. I use the following notational conventions:

- σ_c and σ_t denote any LI $l \in \Lambda$ whose category feature is C or T, respectively,
- the symbol "there" refers to any expletive $l \in \Lambda$ involved in MOM (usually just *there*, but possibly also *it*),
- σ_l denotes any LI which does not 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).

$$\sigma_l \to q_*(\sigma_l) \qquad Merge(q_{c*}(x), q_{t*}(y)) \to q_*(Merge(x,y))$$

$$\sigma_t \to q_t(\sigma_t) \qquad Merge(q_t(x), q_{\{t,*\}}(y)) \to q_t(Merge(x,y))$$

$$\text{there} \to q_{there}(\text{there}) \qquad Merge(q_{there}(x), q_{\{t,*\}}(y)) \to q_t(\Box(y))$$

$$\sigma_c \to q_c(\sigma_c) \qquad Move(q_*(x)) \to q_*(Move(x))$$

$$Move(q_t(x)) \to q_t(\Box(x))$$

Definition 4. *Path Condition* is the lbutt $\mathscr{P}$, where $\Sigma_{\mathscr{P}} := \Omega_{\mathscr{U}}$, $\Omega_{\mathscr{P}} := \Sigma_{\mathscr{U}}$, $Q := \{q_*, q_c, q_{move}\}$, $Q' := \{q_*\}$, and $\Delta_{\mathscr{P}}$ contains the rules below (the same notational conventions apply):

$$\sigma_l \to q_*(\sigma_l) \qquad Merge(q_{c\{c,*\}}(x), q_{move\{c,*\}}(y)) \to q_*(Merge(x,y))$$

$$\sigma_t \to q_*(\sigma_t) \qquad Merge(q_{\{move,*\}}(x), q_{move}(y)) \to q_{move}(Merge(x,y))$$

$$\sigma_c \to q_c(\sigma_c) \qquad Move(q_*(x)) \to q_*(Move(x))$$

$$\square(q_*(x)) \to q_*(Merge(\text{there}, x))$$

$$\square(q_{\{move,*\}}(x)) \to q_{move}(Move(x))$$

Definition 5. *Restore Features* is the non-deterministic transducer computing the inverse of the transduction defined by *Remove Features*.

Let *id* be the identity function over L (which is guaranteed to a be a linear tree transduction). Now we define τ as the composition of *Remove Features*, *Underspecify*, *Path Condition*, *Restore Features*, and *id*. The image of L under τ consists of all trees that are optimal with respect to MOM, and only those.

Acknowledgments

I am greatly indebted to Ed Stabler, Ed Keenan, and Uwe Mönnich for their motivational comments and helpful criticism. Some of the ideas discussed here were previously presented at Formal Grammar 2010, NELS 41, and GLOW 34. I would like to thank the audiences of these conferences, in particular Alex Drummond, Jeffrey Heinz, Tim Hunter, Paul Smolensky, Peter Svenonius, and Charles Yang. The research reported herein was supported by a DOC-fellowship of the Austrian Academy of Sciences.