Chapter 9

An Automata-Theoretic Approach to Minimalism

Gregory M. Kobele, Christian Retoré and Sylvain Salvati
INRIA Futurs, LaBRI (Université de Bordeaux – CNRS)
Bordeaux, France
`kobele|retore|salvati@labri.fr`

Vijay-Shanker et al. (1987) note that many interesting linguistic formalisms can be thought of as having essentially context-free structure, but operating over objects richer than simple strings (sequences of strings, trees, or graphs). They introduce linear context-free rewriting systems (LCFRS's, see also Weir (1988)) as a unifying framework for superficially different such formalisms (like (multi component) tree adjoining grammars, head grammars, and categorial grammars). Later work (Michaelis, 1998) has added minimalist grammars (MGs, see (Stabler, 1997)) to this list. Recently, Fülöp et al. (2004) have introduced multiple bottom-up tree transducers (mbutt), which can be thought of as offering a transductive perspective on LCFRSs. The transductive perspective allows us to view a grammar in one of these grammar formalisms as defining both a set of well-formed derivations, and functions which interpret these derivations as the derived structures (trees, strings, or meanings) they are derivations of. Being explicit about the structure of the derivation, and divorcing it from the construction of the object so derived has two main advantages. First, we may entertain and study the effects of modifications to the structure of the derivational process, such as insisting that a particular operation apply only in case there is an isomorphic subderivation somewhere in the same derivation (for example, in deletion under identity with an antecedent), or other non-local filters on well-formed derivations, without worrying about the kinds of data structures that would be required to support such operations in real-time (as in parsers, for example). Secondly, viewing derivational grammar formalisms in this way makes particularly salient two loci of language theoretic complexity:

1. the set of well-formed derivation structures
2. the transformation from derivation structures to derived structures

Taking this latter perspective, Shieber (2006) shows that TAGs are exactly characterized in terms of monadic macro tree transducers simple in both the input and the parameters (1-MTT$_{si,sp}$) (Engelfriet and Vogler, 1985) acting on a regular tree language (see also Mönnich (1997)).

Minimalist grammars offer a formal perspective on some of the core ideas in Chomsky's minimalist program (Chomsky, 1995) (various extensions to the core formalism have been proposed and investigated; a variant with copying was introduced and studied in (Kobele, 2006)). We show in this paper how, given a minimalist grammar G, to construct a simple, regular, characterization of its well formed derivations. Furthermore, given the close connection between LCFRSs and mbutts, it is straightforward to construct a linear deterministic mbutt which maps derivation trees to the structures they are derivations of. Deterministic mbutts were proven in Fülöp et al. (2004) to be equivalent to deterministic top-down tree transducers with regular look-ahead (dTT^R), and it was conjectured that adding linearity to the mbutt corresponded to restricting the dTT^R to be finite copying. We prove half of this conjecture in the appendix: linear deterministic mbutts (ldmbutt) can be simulated by finite copying deterministic top-down tree transducers with regular look-ahead (dTT^R_{fc}).[1] We obtain thus both a bottom-up and a top-down characterization of the function from minimalist derivations to derived trees. The same construction extends to minimalist grammars with copying simply by removing the finite copying restriction (dTT^R). In other words, the structure languages generated by minimalist grammars with (without) copying are contained in the output languages of (finite copying) tree homomorphisms.

We can immediately conclude that, although the string languages generated by minimalist grammars properly include those generated by TAGs,[2] the same is not true of

[1] Michaelis et al. (2001) have provided a different characterization of the derived trees definable by minimalist grammars (see also Morawietz (2003)). Given a minimalist grammar, they define a regular tree grammar which encodes the operations of an equivalent LCFRS as operation symbols in a lifted signature. From there, they show that one can obtain the desired trees using a monadic second order logic transduction, a MTT simple in the input and the parameters, or a deterministic tree walking automaton. As we think the derivation tree is an interesting object in its own right (as per our introductory comments), we prefer to start from there. Our obtained transducer class is different in non-trivial ways as well, with MSO and simple MTTs able to define transductions which $dTT^R_{(fc)}$s cannot.

[2] MGs were proven to be equivalent to multiple context-free grammars (Seki et al., 1991) in (Michaelis, 1998; Harkema, 2001; Michaelis, 2001). The variant with copying is equivalent to parallel multiple context-free grammars (Seki et al., 1991), see (Kobele, 2006). TAGs

their respective structure languages, as the output languages of deterministic (finite copying) tree transducers are incomparable to those of 1-MTT$_{si,sp}$s (Engelfriet and Maneth, 2000). An example of a derived TAG tree language that is not also generable by an MG is $\{a^n(b^n(e)) : n \geq 1\}$ (as monadic languages which are the output of a regular tree transducer are all recognizable).

Tree transducers can also be used to characterize transformations of trees into non-tree-like structures, such as graphs, or even arbitrary algebras (Bauderon and Courcelle, 1987; Engelfriet, 1994). The idea is to encode elements of the algebra as trees, and to 'decode' the tree $\tau(t)$, for input tree t and transducer τ, into the algebraic object it represents (this is the idea behind the common 'tree-to-string' mappings). For instance, we might interpret the derived objects not as strings, but rather as partially ordered multi-sets, as proposed in Pan (2007), which allows for an elegant statement of otherwise quite difficult to describe (Bobaljik, 1999) word order regularities in languages like Norwegian. Compositionality, the principle that the meaning of an object is determined by the meanings of its immediate parts and their mode of combination, is naturally formulated as a transduction mapping derivation trees to (terms denoting) semantic values. The compositional semantics for minimalist grammars introduced in Kobele (2006) is naturally expressed in terms of a transduction of the same type as that mapping derivations to derived trees (a $dTT^R_{(fc)}$). We present a general method of synchronizing (in the sense of Shieber (1994)) multiple transductions over the same derivation, showing as a result that the form-meaning relations definable by MGs interpreted as per Kobele (2006) can be described as bimorphisms of type $\mathbf{B}(\mathbf{M},\mathbf{M})$ (in the terminology of Shieber (2006)).

The rest of this paper is structured as follows. After some mathematical preliminaries, we introduce minimalist grammars. We then define the derivation tree language of a minimalist grammar, and prove that it is regular. We then introduce multi bottom-up tree transducers, and show that one can therewith transform a minimalist derivation tree into the derived structure it represents a derivation of. Finally, bimorphisms are introduced, and the form-meaning relations generable by minimalist grammars are shown to be contained within the bimorphism class **B**(**M**,**M**) (**M** is the set of unrestricted homomophisms). In the appendix, the linear deterministic multi bottom-up tree transducers used in this paper to establish the above results are shown to be included in the top-down tree transducers with regular look-ahead and finite copying, as conjectured by Fülöp et al. (2005). At the end we include a picture which contains some of the wealth of information on tree languages generated by various grammars and devices.

are equivalent to a proper subclass of multiple context-free grammars (Seki et al., 1991).

9.1 Preliminaries

The set of natural numbers will be denoted by $\mathbb{N}$ and $[n]$ will denote the set $\{1,\ldots,n\}$ with the convention that $[0]$ represents the empty set.

Σ^* is the set of all finite sequences of elements of Σ. Σ^+ is the set of all non-empty such sequences, and ε is the empty sequence. The length of a sequence w is denoted $|w|$, and $|\varepsilon| = 0$. For a non-empty sequence aw, $a \in \Sigma$ is its head and w its tail.

A ranked alphabet Ω is a finite set (also denoted Ω) together with a function $\mathbf{rank} : \Omega \to \mathbb{N}$ assigning to each $\omega \in \Omega$ its rank. The notation $\Omega^{(n)}$, for $n \in \mathbb{N}$, denotes the set $\{\omega \in \Omega : \mathbf{rank}(\omega) = n\}$ of symbols of rank n. Given $\omega \in \Omega^{(n)}$, we sometimes write $\omega^{(n)}$ to remind us that $\mathbf{rank}(\omega) = n$. The set of trees built on a ranked alphabet Ω, noted T_Ω, is the smallest set that contains $\Omega^{(0)}$ and $\omega(t_1,\ldots,t_n)$ iff for all $i \in [n]$, $t_i \in T_\Omega$.

9.2 Minimalist Grammars

An idea common to many grammar formalisms is that natural languages are resource sensitive, in the sense that grammatical operations consume resources when applied. Minimalist grammars implement this idea in terms of *features*, which are deleted or *checked* as operations are applied. Syntactic features come in two varieties: **licensing** features and **selection** features, which are relevant for the grammatical operations of **move** and **merge** respectively. Each feature type has a positive and a negative polarity. The set of licensing features is **lic**, and for $x \in$ **lic**, +x is the positive, and -x the negative polarity feature of type x. The set of selection features is **sel**, and for $x \in$ **sel**, =x is the positive, and x the negative polarity feature of type x. We assume without loss of generality that **lic** and **sel** are disjoint. $\mathbb{F} = \{+x, -x, =y, y : x \in \mathbf{lic}, y \in \mathbf{sel}\}$ is the set of all positive and negative polarity features of all types. An expression $\phi = \phi_0, \phi_1, \ldots, \phi_n$ is a finite sequence of pairs $\phi_i = \langle t_i, l_i \rangle$ of trees t and sequences of features l.[3] The intuition is that the grammatical operations combine trees in various ways based on the features that are associated with these trees. Given an alphabet Σ and a symbol $\varepsilon \notin \Sigma$ we will interpret as the empty string (the set $\Sigma \cup \{\varepsilon\}$ is denoted Σ_ε), the tree components of a minimalist expression have internal nodes labelled with either < or > (indicating that the head of the tree as a whole is the head of the left or right subtree respectively), and leaves labelled with either t (a 'trace') or elements of Σ_ε. These labels form a ranked alphabet $\Omega = \{<^{(2)}, >^{(2)}, t^{(0)}\} \cup \Sigma_\varepsilon$, where each $\sigma \in \Sigma_\varepsilon$ has rank 0. In lexicalized grammar formalisms like MGs, the grammatical operations are held constant across grammars, the locus of variation being confined to different choices of lexical items. A *lexicon* is a finite set

[3] This is the 'chain-based' presentation of MGs (Stabler and Keenan, 2003), but with trees, and not strings, as the derived objects. The possibility of such a representation was first noticed by Michaelis (1998), who used it to prove the containment of the minimalist languages in the MCFGs.

$Lex \subset \Sigma_\varepsilon \times \mathbb{F}^+$. The grammatical operations **move** and **merge**, common to all minimalist grammars, are defined as per the following. The unary operation **move** is defined on an expression $\phi_0, \ldots, \phi_n$ just in case the head of the feature sequence of ϕ_0 is a positive polarity licensing feature type $\texttt{+x}$, and there is exactly one ϕ_i the head of whose sequence is the corresponding negative feature $\texttt{-x}$ (the requirement that ϕ_i be unique is called the SMC, and results in an upper bound of $|\mathbf{lic}|+1$ on the length of useful expressions). The definition of **move** is given in two cases, as per whether the moving element has exactly one (**move1**), or more than one (**move2**) feature in its feature sequence. The tree denoted by $f(t_1,t_2)$ is $<(t_1,t_2)$ if $t_1 \in \Sigma_\varepsilon$, and is $>(t_2,t_1)$ otherwise. For l_i non-empty,

$$\mathbf{move1}(\langle t_0, \texttt{+x}l_0\rangle, \ldots, \langle t_i, \texttt{-x}\rangle, \ldots, \phi_n) = \langle f(t_0,t_i), l_0\rangle, \ldots, \phi_n$$

$$\mathbf{move2}(\langle t_0, \texttt{+x}l_0\rangle, \ldots, \langle t_i, \texttt{-x}l_i\rangle, \ldots, \phi_n) = \langle f(t_0,\texttt{t}), l_0\rangle, \ldots, \langle t_i, l_i\rangle, \ldots, \phi_n$$

The binary operation **merge** is defined on expressions ϕ and ψ just in case the head of the feature sequence of ϕ_0 is a positive polarity selection feature $\texttt{=x}$, and the head of the feature sequence of ψ_0 is the corresponding negative feature $\texttt{x}$. As before, we split the definition of **merge** into two cases, based on whether the feature sequence of ψ_0 contains exactly one (**merge1**) or more than one (**merge2**) feature. For l'_0 non-empty,

$$\mathbf{merge1}(\langle t_0, \texttt{=x}l_0\rangle, \phi_1, \ldots, \phi_m;\ \langle t'_0, \texttt{x}\rangle, \psi_1, \ldots, \psi_n) = \langle f(t_0,t'_0), l_0\rangle, \phi_1, \ldots, \phi_m, \psi_1, \ldots, \psi_n$$

$$\mathbf{merge2}(\langle t_0, \texttt{=x}l_0\rangle, \phi_1, \ldots, \phi_m;\ \langle t'_0, \texttt{x}l'_0\rangle, \psi_1, \ldots, \psi_n) = \langle f(t_0,\texttt{t}), l_0\rangle, \phi_1, \ldots, \phi_m, \langle t'_0, l'_0\rangle, \psi_1, \ldots, \psi_n$$

Given an alphabet Σ, a minimalist grammar G over Σ is given by its set of features $\mathbb{F}$, a lexicon *Lex*, and a designated feature $\texttt{c} \in \mathbb{F}$ (the type of sentences). The expressions generated by a minimalist grammar $G = \langle \mathbb{F}, Lex, \texttt{c}\rangle$ are those in $CL(Lex) = \bigcup_{n\in\mathbb{N}} CL_n(Lex)$, where[4]

$$\begin{aligned} CL_0(Lex) &= Lex \\ CL_{n+1}(Lex) &= CL_n(Lex) \\ &\quad \cup \{\mathbf{move}(\phi) : \phi \in CL_n(Lex)\} \\ &\quad \cup \{\mathbf{merge}(\phi,\psi) : \phi,\psi \in CL_n(Lex)\} \end{aligned}$$

An expression $\phi = \phi_0, \ldots, \phi_n$ is complete iff $n = 0$. The structure language $S(G) = \{t : \langle t, \texttt{c}\rangle \in CL(Lex)\}$ generated by G is the set of tree components of complete expressions whose feature sequence component is the designated feature $\texttt{c}$.

[4] It is implicitly assumed that the arguments presented to the generating functions are restricted to those in their domains.

9.3 Derivations as trees

Given a minimalist grammar over Σ, $G = \langle \mathbb{F}, Lex, \texttt{c}\rangle$, its derivation trees are defined to be the terms over the ranked alphabet $\Gamma = \{\mathbf{mrg}^{(2)}, \mathbf{mv}^{(1)}\} \cup Lex$, where the elements of *Lex* have rank 0. A derivation tree $t \in T_\Gamma$ is *a derivation of* an expression ϕ just in case $\phi = h(t)$, where h maps lexical items to themselves, and $h(\mathbf{mv}(t)) = \mathbf{move}(h(t))$ and $h(\mathbf{mrg}(t_1,t_2)) = \mathbf{merge}(h(t_1),h(t_2))$. As the functions **merge** and **move** are partial, so is h. We can identify the set of *convergent* (well-formed) derivation trees with the domain of h.

The first question we ask is as to the language theoretic complexity of the set of well-formed derivation trees of complete expressions. We will show (by exhibiting the automaton) that this set is the language accepted by a bottom-up tree automaton; in other words, a regular tree language. A bottom-up tree automaton (BA) is a structure $A = \langle Q, Q_f, \Sigma, \delta\rangle$, where Q is a finite set of states, $Q_f \subseteq Q$ the set of final states, Σ is a ranked alphabet, and $\delta = (\delta_\sigma)_{\sigma\in\Sigma}$ is a family of partial functions $\delta_\sigma : Q^{\mathbf{rank}(\sigma)} \to 2^Q$ from $\mathbf{rank}(\sigma)$-tuples of states to sets of states. If for every $\sigma^{(n)} \in \Sigma$, and for every $q_1, \ldots, q_n \in Q$, $|\delta_\sigma(q_1,\ldots,q_n)| \leq 1$ then A is deterministic, and we write $\delta_\sigma(q_1,\ldots,q_n) = q$ for $\delta_\sigma(q_1,\ldots,q_n) = \{q\}$. For a term $\sigma^{(n)}(t_1,\ldots,t_n) \in T_\Sigma$, $\delta(\sigma(t_1,\ldots,t_n)) = \bigcup\{\delta_\sigma(q_1,\ldots,q_n) : q_i \in \delta(t_i)\}$. A term $t \in T_\Sigma$ is accepted by A just in case $\delta(t) \cap Q_f$ is non-empty.

Theorem 9.3.1. *For $G = \langle \mathbb{F}, Lex, \texttt{c}\rangle$ a minimalist grammar over an alphabet Σ, and for $l \in \mathbb{F}^+$, the set of convergent derivation trees of complete expressions of type l is a regular tree language.*

Proof. We construct a deterministic bottom up tree automaton $A_G = \langle Q, Q_f, \Gamma, \delta\rangle$ which recognizes just the convergent derivations in T_Γ of complete expressions of type l. Any set accepted by such an automaton is regular, whence the conclusion. The states of our automaton will keep track of the featural components of the expression $h(t)$ that the derivation tree t is a derivation of. To bring out the logical structure of the feature calculus (and thereby simplify the statement of the transition function), instead of working with arbitrary sequences of feature sequences (the right projection of minimalist expressions) we represent the features had by an expression ϕ as an $n+1$-ary sequence of feature sequences, with $n = |\mathbf{lic}|$ (recall that the SMC condition on **move** ensures that no expression that is part of a convergent derivation of a complete expression has more than one subpart ϕ_i with feature sequence beginning $\texttt{-x}$, for any $x \in \mathbf{lic}$). Moreover, an arbitrary but fixed enumeration of **lic** allows us to denote licensing feature types with positive integers (thus $\texttt{+1}$ denotes a positive polarity feature of the first licensing feature type), and we require that the i^{th} component of our states, if non-empty, contain a feature sequence beginning with $\texttt{-i}$. Formally, for $\texttt{suf}(Lex) := \{\beta : \langle \sigma, \alpha\beta\rangle \in Lex\}$ the set of suffixes of lexical feature sequences, we define

our set of states such that

$$Q := \{\langle s_0,\ldots,s_n\rangle : s_0,\ldots,s_n \in \mathtt{suf}(Lex) \text{ and for } 1 \le i \le n \text{ either } s_i = \varepsilon \text{ or } s_i = \mathtt{-i}\alpha\}$$

The set of final states Q_f is the singleton $\{\langle l,\varepsilon,\ldots,\varepsilon\rangle\}$ It remains to describe the action of the transition function on states. To make the description of the results of these functions easier, we define the partial binary operation over feature sequences $\oplus$ ('sum') which is defined just in case at least one of its arguments is ε, and returns its non-empty argument if one exists, and ε otherwise. We extend $\oplus$ to a function which takes a state $q = \langle s_0,\ldots,s_i,\ldots\rangle$ and a feature sequence s and returns q if $s = \varepsilon$ and $\langle s_0,\ldots,(s_i \oplus s),\ldots\rangle$ if $s = \mathtt{-i}s'$ (otherwise, $\oplus$ is undefined). The transition function $\delta_{\mathbf{mv}}$ is defined on a state $q = \langle s_0,\ldots,s_n\rangle$ just in case the head of the sequence of features in the initial position of is a positive polarity licensing feature (`+i`), the head of the feature sequence in the i^{th} position is the corresponding negative polarity licensing feature (`-i`), and if the tail of the feature sequence in the i^{th} position is non-empty and begins with `-j`, then the j^{th} position is empty. If defined, the result is identical to q, except that the matching i^{th} licensing features are deleted, and the remainder of the feature sequence in the i^{th} array position is moved to the j^{th} array position if it begins with `-j`. Formally,

$$\delta_{\mathbf{mv}}(\langle \mathtt{+i}s_0,\ldots,\mathtt{-i}s_i,\ldots\rangle) = \langle s_0,\ldots,\varepsilon,\ldots\rangle \oplus s_i$$

The transition function $\delta_{\mathbf{mrg}}$ applies to a pair of states just in case the following three conditions are met. First, the heads of the initial feature sequence of the two states must be positive and negative polarity features of the same selection feature type, respectively. Second, whenever a non-initial feature sequence of the first state is non-empty, the corresponding feature sequence in the second state must be empty. Finally, if the tail of the initial feature sequence of the second state begins with `-j`, then the j^{th} position of both states must be empty. If defined, $\delta_{\mathbf{mrg}}(q_1,q_2)$ is the state whose initial component is the tail of the initial component of q_1, whose j^{th} component is the sum of the tail of the initial component of q_2 with the j^{th} components of both input states, and whose non-initial components are the sum of the corresponding non-initial components in q_1 and q_2. Formally,

$$\delta_{\mathbf{mrg}}(\langle \mathtt{=f}s_0,\ldots,s_i,\ldots\rangle, \langle \mathtt{f}s'_0,\ldots,s'_i,\ldots\rangle) = \langle s_0,\ldots,(s_i \oplus s'_i),\ldots\rangle \oplus s'_0$$

Finally, for each lexical item $\langle\sigma,l\rangle$, $\delta_{\langle\sigma,l\rangle}$ is the constant function that outputs the state with initial component l, and all other components ε. Formally,

$$\delta_{\langle\sigma,l\rangle}(\langle\sigma,l\rangle) = \langle l,\varepsilon,\ldots,\varepsilon\rangle$$

A simple induction on derivation trees proves the correctness of the automaton. The only slightly tricky bit stems from the fact that the automaton in effect enforces the SMC at each step, whereas the minimalist grammars 'wait' until a **move** step. This is harmless as once an expression is generated which has more than one component with the same initial `-i` feature it can never be 'rescued' and turned into a complete expression. □

As a special case we obtain

Corollary 9.3.2. *For any MG $G = \langle \mathbb{F}, Lex, \mathtt{c}\rangle$, the set of derivation trees of sentences (complete expressions of category* `c`*) is regular.*

9.4 Interpreting derivations

The picture of minimalist grammars with which we began conflates the structure of the feature calculus with the process of tree assembly. We have seen that by factoring out the tree-building operations from the syntactic feature manipulation, we are left with a simple and elegant system, and that the structure of the feature calculus is underlyingly regular. We can think of the syntactic calculus as delivering blueprints for building trees. We now know that these blueprints themselves have a simple regular structure, but what is left to determine is the complexity of building trees from blueprints.

We will extend the bottom-up automaton from the previous section (which manipulated sequences of feature sequences) so as to allow it to build trees. In minimalist expressions $\phi = \phi_0,\ldots,\phi_m$, each tree t_i is paired with its syntactic features s_i directly. This makes the order of occurrence of the ϕ_is irrelevant. In contrast, in our automata, features are used in the description of states, and thus are dissociated from their trees. Accordingly, we make the objects derived during a derivation $n+1$-ary sequences of trees over the ranked alphabet $\Omega = \{<^{(2)}, >^{(2)}, \mathtt{t}^{(0)}\} \cup \Sigma_\varepsilon$. The connection between a tree and its feature sequence is established by the invariant that the i^{th} component of a state represents the features of the i^{th} tree in a sequence. There are n^2+2n+1 basic operations on $n+1$-ary sequences of trees: $\mathbf{m}^{(2)}$, $\mathbf{m_j}^{(2)}$, $\mathbf{v_i}^{(1)}$, and $\mathbf{v_{i,j}}^{(1)}$, for $1 \le i,j \le n$. These operations form an algebra $\mathcal{S}$ over the carrier set $S = \{\langle t_0,\ldots,t_n\rangle : t_0,\ldots,t_n \in T_\Omega\}$ of $n+1$-ary tree sequences. Intuitively, the operations on tree sequences are indexed to particular cases of the $\delta_{\mathbf{mv}}$ and $\delta_{\mathbf{mrg}}$ functions, and derivations in the syntactic calculus then control tree sequence assembly (as shown in figure 9.1). The operations are defined as per the following, where $t_1 \oplus t_2$ is defined iff at least one of t_1 and t_2 is $\mathtt{t}$, in which case it returns the other one.

$$\mathbf{v_i}(\langle t_0,\ldots,t_i,\ldots\rangle) = \langle f(t_0,t_i),\ldots,\mathtt{t},\ldots\rangle$$

$$\mathbf{v_{i,j}}(\langle t_0,\ldots,t_i,\ldots,t_j,\ldots\rangle) = \langle f(t_0,\mathtt{t}),\ldots,\mathtt{t},\ldots,(t_j \oplus t_i),\ldots\rangle$$

$$\mathbf{m}(\langle t_0,\ldots,t_i,\ldots\rangle, \langle t'_0,\ldots,t'_i,\ldots\rangle) = \langle f(t_0,t'_0),\ldots,(t_i \oplus t'_i),\ldots\rangle$$

operation	case of $\delta_{\mathbf{mv}}/\delta_{\mathbf{mrg}}$
$\mathbf{v_i}$	$s_i = \varepsilon$
$\mathbf{v_{i,j}}$	$s_i = \texttt{-j}l$
$\mathbf{m}$	$s'_0 = \varepsilon$
$\mathbf{m_j}$	$s'_0 = \texttt{-j}l$

Figure 9.1: Operations on tree sequences and the syntactic operations they are associated with

$$\mathbf{m_j}(\langle t_0,\ldots,t_j,\ldots\rangle, \langle t'_0,\ldots,t'_j,\ldots\rangle) = \langle f(t_0,\mathtt{t}),\ldots,(t_j \oplus t'_j)\oplus t'_0,\ldots\rangle$$

Each state $q = \langle s_0,\ldots,s_n\rangle$ is associated with an $n+1$-tuple of trees $\langle t_0,\ldots,t_n\rangle$. We would like the states to be put together in accord with the transition function δ from the proof of theorem 9.3.1, and the tuples of trees in accord with the operations in figure 9.1. Thus, we would like to map $\mathbf{mrg}(u,v)$, where u is mapped to $q(t_0,\ldots,t_n)$ and v to $q'(t'_0,\ldots,t'_n)$, to $\delta_{\mathbf{mrg}}(q,q')(\mathbf{m}(\langle t_0,\ldots,t_n\rangle, \langle t'_0,\ldots,t'_n\rangle))$ if the first component of q' is of length one, and to $\delta_{\mathbf{mrg}}(q,q')(\mathbf{m_j}(\langle t_0,\ldots,t_n\rangle, \langle t'_0,\ldots,t'_n\rangle))$ if the tail of the first component of q' begins with $\texttt{-j}$. This intuitive picture is very close to the multi bottom-up tree transducer model introduced in (Fülöp et al., 2004). A multi bottom-up tree transducer is a tuple $M = (Q,\Sigma,\Delta,root,f,R)$ where:

1. Q is a ranked alphabet, *the states*, with $Q^{(0)} = \emptyset$
2. Σ and Δ are ranked alphabets, respectively the *input alphabet* and the *output alphabet*
3. *root* is a unary symbol called the *root*
4. f is the final 'state'. Q, $\Sigma\cup\Delta$, $\{root\}$, and $\{f\}$ are pairwise disjoint sets
5. R is the set of *rules* which are of one of the two forms below, for $\sigma\in\Sigma^{(n)}$, $q_k\in Q^{(r_k)}$, and $t_l \in T_\Delta(\{y_{1,1},\ldots,y_{1,r_1},\ldots,y_{n,1},\ldots,y_{n,r_n}\})$ and $q\in Q^{(n)}$ and $t\in T_\Delta(X_n)$

$$\sigma(q_1(y_{1,1},\ldots,y_{1,r_1}),\ldots,q_n(y_{n,1},\ldots,y_{n,r_n})) \to q_0(t_1,\ldots,t_{r_0})$$

$$root(q(x_1,\ldots,x_n)) \to f(t)$$

An mbutt is *linear* just in case each of the variables occur *at most once* in *at most one* of the output trees. It is *deterministic* just in case there are no two productions with the same left hand sides.

Theorem 9.4.1. *For every minimalist grammar $G = \langle\mathbb{F},Lex,\mathtt{c}\rangle$, there is a linear deterministic multi-bottom up tree transducer M_G such that for $L(A_G)$ the set of derivations of complete expressions of category* c*, $M_G(L(A_G)) = S(G)$.*

Proof. The states of M_G are triples of states from our bottom-up tree automaton, our tree sequence algebra operations, and a boolean value which encodes the result of the test for the function f (whether or not the first tree is a symbol from Σ_ε). Each state has the same arity, $|\mathbf{lic}|+1$. Our rules R include[5]

1. for $q = \delta_{\mathbf{mv}}(q_1)$, $\rho\in\{\mathbf{v_i},\mathbf{v_{i,j}}\}$ such that the condition in figure 9.1 is satisfied, and $\langle t_0,\ldots,t_n\rangle = \rho(\langle y_{1,1},\ldots,y_{1,n+1}\rangle)$,

$$\mathbf{mv}(\langle q_1,\rho_1,b_1\rangle(y_{1,1},\ldots,y_{1,n+1})) \to \langle q,\rho,0\rangle(t_0,\ldots,t_n)$$

2. for $q = \delta_{\mathbf{mrg}}(q_1,q_2)$, $\rho\in\{\mathbf{m},\mathbf{m_i}\}$ such that the condition in figure 9.1 is satisfied, and $\langle t_0,\ldots,t_n\rangle = \rho(\langle y_{1,1},\ldots,y_{1,n+1}\rangle,\langle y_{2,1},\ldots,y_{2,n+1}\rangle)$,

$$\mathbf{mrg}(\langle q_1,\rho_1,b_1\rangle(y_{1,1},\ldots,y_{1,n+1}), \langle q_2,\rho_2,b_2\rangle(y_{2,1},\ldots,y_{2,n+1})) \to \langle q,\rho,0\rangle(t_0,\ldots,t_n)$$

3. $\langle\sigma,l\rangle \to \langle q,\rho,1\rangle(\sigma,\mathtt{t},\ldots,\mathtt{t})$ just in case $q = \delta_{\langle\sigma,l\rangle}(\langle\sigma,l\rangle)$
4. $root(\langle q,\rho,b\rangle(x_1,\ldots,x_{n+1})) \to f(x_1)$ just in case $q = \langle\mathtt{c},\varepsilon,\ldots,\varepsilon\rangle$

Again, a simple induction on derivation trees suffices to establish the correctness of the construction. □

Given the construction of the ldmbutt in theorem 9.4.1, it is clear that we could just as well have chosen different operations over different $n+1$ tuples of objects (Kobele (2006) provides an example of such). Additionally, we can use the very same feature calculus to simultaneously control different operations over different algebras. A synchronization of two dmbutts M and M' is a triple $\langle M,M',C\rangle$ where $C\subseteq R\times R'$ is the control set, which serves to specify which transitions in M are allowed to be used with which productions in M'. The relation defined by such an object is

$$\{\langle u,v\rangle : \exists t\in T_\Sigma.\exists c\in C^*.\ t\vdash_M^{\pi_1(c)} u \wedge t\vdash_{M'}^{\pi_2(c)} v\}$$

where π_i is the i^{th} projection function extended over sequences in the obvious way, $u\vdash_M^{aw} v$ just in case $u\vdash_M^a v'$ and $v'\vdash_M^w v$, and $u\vdash_M^a v$ just in case a is a production in M, u rewrites to v in a single step using a, and a is applied to the left-most rewritable node in u.

[5] The resolution of the operation $\oplus$ in the definition of ρ must be done by the states q_1 and q_2. As an empty component in a state is shadowed by a trace in a tree sequence, this is merely a notational inconvenience.

This is (a restriction to a particular transducer type of) a generalization of the model of synchronous tree adjoining grammars, as developed by Shieber and Schabes (1990), and thus can be used, for example, to model the syntax-semantics interface (Nesson and Shieber, 2006). Shieber (2006) investigates the complexity of the form-meaning relationships definable by synchronous TAGs by situating them within the context of bimorphisms. A bimorphism is a triple $B = \langle \Phi, L, \Psi \rangle$, where L is a recognizable tree language and Φ and Ψ are homomorphisms; the relation it defines is $L(B) = \{\langle \Phi(t), \Psi(t) \rangle : t \in L\}$. Given classes $\mathbf{H_1}$ and $\mathbf{H_2}$ of homomorphisms, $\mathbf{B}(\mathbf{H_1}, \mathbf{H_2})$ denotes the class of bimorphisms $\langle h_1, L, h_2 \rangle$ where $h_i \in \mathbf{H_i}$ and L is recognizable. Shieber proves that synchronous TAGs define exactly those relations definable by bimorphisms where the homomorphisms are one state monadic macro tree transducers simple in the input and parameters.

The following theorem is an easy consequence of a result in Fülöp et al. (2004).

Theorem 9.4.2. *The relation defined by a synchronization $\langle M, M', C \rangle$ of dmbutts M and M' is in the bimorphism class $\mathbf{B}(\mathbf{M}, \mathbf{M})$, where $\mathbf{M}$ is the class of unrestricted homomorphisms. It is in the class $\mathbf{B}(\mathbf{FC}, \mathbf{FC})$, where $\mathbf{FC}$ is the class of finite copying homomorphisms, if M and M' are linear.*

Proof. By moving to the expanded alphabet $\Sigma \times R \times R'$, we can find new dmbutts $M_\star$ and $M'_\star$ such that the set $\{\langle M_\star(t), M'_\star(t) \rangle : t \in T_{\Sigma \times R \times R'}\}$ is the very same relation as defined by $\langle M, M', C \rangle$ (we essentially encode the control information into the vocabulary itself). By theorem 4.4 in Fülöp et al. (2004), we can find an equivalent dTT^R for any dmbutt. It is well-known that regular look-ahead and states can be encoded into a regular set of trees (Engelfriet, 1977), and therefore for any dTT^R T and regular language L we can find a homomorphism h and regular language L_h such that $T(L) = h(L_h)$. Thus, from M and M' over T_Σ, we move to $M_\star$ and $M'_\star$ over $T_{\Sigma \times R \times R'}$, and from there we obtain $T_\star$ and $T'_\star$, whence we finally arrive at homomorphisms $h_\star$ and $h'_\star$. By the result in the appendix, $h_\star$ ($h'_\star$) is finite copying if M (M') is linear. □

9.5 Conclusion

In light of our transductive characterization of minimalist grammars, what seems the core of the minimalist grammar framework is the underlying feature calculus, and the n-tuples of terms that are therewith naturally controllable. The cases of the generating functions (**merge1**,**merge2**,...) that were introduced at the beginning are now revealed to be gerrymanderings of the feature calculus to support the particular mode of manipulating expressions qua minimalist trees. Different modes of expression manipulation, or different choices of expressions to manipulate, might well have drawn different lines in the sand. This perspective allows us to consider the *relations* that the minimalist feature calculus makes definable. Situating natural langauge formalisms in the context of bimorphisms provides an elegant and principled way of measuring and comparing their 'strong generative capacity'—the kinds of form-meaning relations the formalism can define. We have seen that all of the relations definable by synchronous minimalist grammars are naturally expressible as bimorphisms where the component maps are simple tree-to-tree homomorphisms. Our characterization is still loose. We must leave it for future work to determine a tighter description of the relations naturally definable by minimalist grammars.

Appendix

In this appendix we show the inclusion of the relations definable by linear deterministic multi bottom-up tree transducers in those definable by single use deterministic top-down tree transducers with regular look-ahead (dTT^R_{su}) which are known to be equivalent to deterministic top-down tree transducers with regular look-ahead with finite copying (dTT^R_{fc}) (Engelfriet and Maneth, 1999). First, some definitions.

Given Σ and Δ two ranked alphabets, we define $\Sigma \cup \Delta$ to be the ranked alphabet such that $(\Sigma \cup \Delta)^{(n)} = \Sigma^{(n)} \cup \Delta^{(n)}$. A set A is made into a ranked alphabet $R(A)$ such that $R(A)^{(0)} = A$ and $R(A)^{(k)} = \emptyset$ when $k > 0$. In particular we write $T_\Omega(A)$ for $T_{\Omega \cup R(A)}$.

We describe tree substitution with a set of indexed input variables $X = \{x_k : k \in \mathbb{N} \wedge k > 0\}$ and also a set of double indexed input variables $Y = \{y_{i,j} : i \in \mathbb{N} \wedge i > 0 \wedge j \in \mathbb{N} \wedge j > 0\}$. The set X_n will denote $\{x_k : k \in [n]\}$ and the set $Y_{n,\langle r_1,\ldots,r_n \rangle}$ will denote $\{y_{k,l} : k \in [n] \wedge l \in [r_k]\}$. Given $t \in T_\Sigma(X_n)$ (resp $T_\Sigma(Y_{n,\langle r_1,\ldots,r_n \rangle})$) and for $k \in [n]$ $t_k \in T_\Sigma$ (resp $k \in [n]$, $l \in [r_k]$ and $t_{k,l} \in T_\Sigma$), we write $t[t_1, \ldots, t_n]$ (resp $t[t_{1,[r_1]}, \ldots, t_{n,[r_n]}]$) for the result of replacing every occurrence of x_k (resp $y_{k,l}$) in t by t_k (resp $t_{k,l}$) for all $k \in [n]$ (resp $k \in [n]$ and $l \in [r_k]$). Given $\sigma \in \Sigma^{(r_k)}$ and a family $(t_{k,l})_{l \in [r_k]}$, we abbreviate $\sigma(t_{k,1}, \ldots, t_{k,r_k})$ to $\sigma(t_{k,[r_k]})$ (a particular case is when $t_{k,l} = y_{k,l}$). We also assume z to be a variable that is neither in X nor in Y and we use to define contexts. A context of T_Σ is an element C of $T_\Sigma(\{z\})$ such that z occurs exactly once in C. Given a t we write $C[t]$ the tree obtained by replacing the occurrence of z in C by t. Contexts will alway be written using capital C with indicies, exponents, primes, ...

Definition 9.5.1. *A top-down tree transducer with regular look-ahead (TT^R for short) is a tuple $M = (Q, \Sigma, \Delta, q_0, R, P, \delta)$ where*

1. *Q is a ranked alphabet of* states *such that $Q^k = \emptyset$ for every $k \in \mathbb{N} \backslash \{1\}$.*
2. *Σ and Δ are ranked alphabets, repectively, the* input alphabet *and the* output alphabet. *Q and $\Sigma \cup \Delta$ are disjoint.*

3. q_0 *is an element of* Q*, the* initial state.
4. (P,Σ,δ) *is a deterministic BA (the final states are unnecessary and will be suppressed).*
5. R *a subset of* $\bigcup_{\Sigma^{(n)}\neq\emptyset} Q\times\Sigma^{(n)}\times T_\Delta(\{q(x_i):q\in Q\wedge x_i\in[n]\})\times P^n$*, the* rules.

As usual, the rule $(q,\sigma^{(n)},t,(p_1,\ldots,p_n))$ of a $tdTT^R$ will be written

$$\langle q(\sigma(x_1,\ldots,x_n))\to t,p_1,\ldots,p_n\rangle.$$

A $tdTT^R$, $M=(Q,\Sigma,\Delta,q_0,R,P,\delta)$, is said to be deterministic whenever given rules

1. $\langle q(\sigma(x_1,\ldots,x_n))\to t,p_1,\ldots,p_n\rangle$ and
2. $\langle q(\sigma(x_1,\ldots,x_n))\to t',p_1,\ldots,p_n\rangle$

in R then $t=t'$. The class of dTT^R that are deterministic will be written dTT^R.

A TT^R, $M=(Q,\Sigma,\Delta,q_0,R,P,\delta)$, defines a relation on $T_{\Sigma\cup\Delta\cup Q}$. Given $t,t'\in T_{\Sigma\cup\Delta\cup Q}$, we write $t\to_M t'$ if there is C, a context of $T_{\Sigma\cup\Delta\cup Q}$, $\langle q(\sigma(x_1,\ldots,x_n))\to v,p_1,\ldots,p_n\rangle\in R$ and $(t_k)_{k\in[n]}$ verifying:

1. $t=C[q(\sigma(t_1,\ldots,t_n))]$
2. for all $k\in[n]$, $t_k\in\mathcal{L}(P,p_k)$
3. $t'=C[v[t_1,\ldots,t_n]]$.

The reflexive and transitive closure of $\to_M$ is written $\Rightarrow_M$, and the relation that M defines between T_Σ and T_Δ is

$$\mathcal{R}_M=\{(t,t'):t\in T_\Sigma\wedge t'\in T_\Delta\wedge q_0(t)\Rightarrow_M t'\}.$$

We now introduce the notion of strongly single use and single use deterministic top-down transduction that has been introduced in Engelfriet and Maneth (1999).

Definition 9.5.2. *Let* $M=(Q,\Sigma,\Delta,q_0,R,P,\delta)$ *be a* dTT^R *and* $\overline{Q}$ *be a nonempty subset of* Q*. M is said* strongly single use with respect to $\overline{Q}$*, if for all* $q,q'\in\overline{Q}$ *and all two rules of* R *:*

1. $\langle q(\sigma(x_1,\ldots,x_n))\to v,p_1,\ldots,p_n\rangle$
2. $\langle q'(\sigma(x_1,\ldots,x_n))\to w,p_1,\ldots,p_n\rangle$

the existence of contexts C *and* C'*,* $q''\in Q$ *and* $j\in[n]$ *such that* $v=C[q''(x_j)]$ *and* $w=C'[q''(x_j)]$ *implies* $q=q'$ *and* $C=C'$*.*

If M is strongly single use with respect to Q the M is said strongly single use.

Definition 9.5.3. *Let* $M=(Q,\Sigma,\Delta,q_0,R,P,\delta)$ *be a* dTT^R*. M is said* single use *if there is a partition* Π *of* Q *and a collection of mappings* $(\mathcal{T}_{\sigma,\langle p_1,\ldots,p_n\rangle}:\Pi\times[n]\to\Pi:\sigma\in\Sigma^{(n)},p_1,\ldots,p_n\in P)$ *such that:*

1. *for all* $\overline{Q}\in\Pi$*, M is strongly single use with respect to* $\overline{Q}$ *and*
2. *for all* $\langle q(\sigma(x_1,\ldots,x_n))\to v,p_1,\ldots,p_n\rangle\in R$ *with* $q\in\overline{Q}\in\Pi$*, if there is an occurrence of* $q'(x_i)$ *in* v *then* $q'\in\mathcal{T}_{\sigma,\langle p_1,\ldots,p_n\rangle}(\overline{Q},i)$*.*

The partition Π *is called a* su partition *for M and* $\mathcal{T}$ *is called a* collection of su mapping for M*. We will write* dTT^R_{su} *to denote the class of* dTT^R *that are single use.*

We now define the relation computed by multi bottom-up tree transduction.

A mbutt $M=(Q,\Sigma,\Delta,root,q_f,R)$ defines a relation $\to_M$ on the trees of $T_{Q\cup\Sigma\cup\Delta\cup\{root,q_f\}}$. Given $t,t'\in T_{Q\cup\Sigma\cup\Delta\cup\{root,q_f\}}$, we have $t\to_M t'$ if there is C a context of $T_{Q\cup\Sigma\cup\Delta\cup\{root,q_f\}}$ verifying one of the two following properties:

1. $t=C[\sigma(q_1(t_{1,[r_1]}),\ldots,q_n(t_{n,[r_n]}))]$, $t'=C[q_0(t_1,\ldots,t_{r_0})[t_{1,[r_1]},\ldots,t_{n,[r_n]}]]$ and
$$\sigma(q_1(y_{1,[r_1]}),\ldots,q_n(y_{n,[r_n]}))\to q_0(t_1,\ldots,t_{r_0})\in R$$
2. $t=C[root(q(t_1,\ldots,t_n))]$, $t'=C[q_f(v[t_1,\ldots,t_n])]$ and
$$root(q(x_1,\ldots,x_n))\to q_f(v)\in R$$

The reflexive and transitive closure of $\to_M$ is denoted by $\Rightarrow_M$. M defines a relation between T_Σ and T_Δ, $\mathcal{R}_M=\{(t,t')\in T_\Sigma\times T_\Delta:root(t)\Rightarrow_M q_f(t')\}$.

A mbutt, $M=(Q,\Sigma,\Delta,root,q_f,R)$, is called *deterministic* whenever

1. $\sigma(q_1(y_{1,[r_1]}),\ldots,q_n(y_{n,[r_n]}))\to q_0(t_1,\ldots,t_{r_0})\in R$ and
2. $\sigma(q_1(y_{1,[r_1]}),\ldots,q_n(y_{n,[r_n]}))\to q'_0(t'_1,\ldots,t'_{r'_0})\in R$

imply $q_0=q'_0$ and for all $k\in[r_0]$, $t_k=t'_k$.

Now that we have defined all the necessary notions, we prove that the classes of transduction realized by dTT^R_{su} include those defined by ldmbutts. In Fülöp et al. (2004), it is shown that dTT^R and dmbutts define the same class of transduction. We here prove the transductions defined by ldmbutts can be defined by dTT^R_{su}; this proof uses the same construction as in lemma 4.1 in Fülöp et al. (2004) and we thus only have to prove that when this construction is used on a ldmbutt it ouputs a dTT^R_{su}.

Let $M=(Q,\Sigma,\Delta,root,q_f,R)$ be a ldmbutt and $A=(Q,\Sigma,\delta)$ be the *dBA* underlying M. We construct the dTT^R $T=(Q',\Sigma,\Delta,p_0,R',Q,\delta)$ as follows:

1. $Q'=\{p_0\}\cup\{\langle q,j\rangle:q\in Q^{(n)}\wedge j\in[n]\}$
2. R' is the smallest set of rules verifying:
   (a) if $\sigma(q_1(y_{1,[r_1]}),\ldots,q_n(y_{n,[r_n]}))\to q_0(t_1,\ldots,t_{r_0})\in R$ then for all $j\in[r_0]$,
   $$\langle\langle q,j\rangle(\sigma(x_1,\ldots,x_n))\to t_j[t_{1,[r_1]},\ldots,t_{n,[r_n]}],q_1,\ldots,q_k\rangle\in R'$$
   with for $k\in[n]$ and $l\in[r_k]$, $t_{k,l}=\langle q_k,l\rangle(x_k)$.

(b) if $root(q(x_1,\ldots,x_n) \to q_f(t)) \in R$ and for all $k \in [n]$ we have that there is, in R' a rule of the form

$$\langle\langle q,k\rangle(\sigma(x_1,\ldots,x_k))t'_k, q_1,\ldots,q_n\rangle$$

then $\langle p_0(\sigma(x_1,\ldots,x_n)) \to t[t'_1,\ldots,t'_n], q_1,\ldots q_n\rangle \in R'$

Fülöp et al. (2004) proves that $\mathcal{R}_T = \mathcal{R}_m$ and that T is indeed a dTT^R. We just have to prove that from the fact that M is linear, T effects a single use transduction.

Although T is not itself single use (the start state p_0 does not satisfy the definition), we will prove that the transducer T' naturally obtained from T by suppressing p_0 **is**. Since p_0 is used only once at the very beginning of any transduction performed by T, it follows that T is finite copying, and can thus be turned into a single use transducer (Engelfriet and Maneth, 1999). To prove that T' is single use we need to find an *su* partition Π for T'. We define Π to be $(\Pi_q)_{q\in Q}$ with $\Pi_q = \{\langle q,i\rangle | i \in [rank(q)]\}$. An element of Π corresponds to the set of states of Q' that are defined from a unique state of Q.

We now first prove that for a given $q \in Q$, T' is strictly single use with respect to Π_q. Suppose that the rules $\langle\langle q,i\rangle(\sigma(x_1,\ldots,x_n)) \to v, q_1,\ldots,q_n\rangle$ and $\langle\langle q,j\rangle(\sigma(x_1,\ldots,x_n)) \to w, q_1,\ldots,q_n\rangle$ are in R', because M is deterministic there is in R a unique rule of the form

$$\sigma(q_1(y_{1,1},\ldots,y_{1,r_1}),\ldots,q_n(y_{n,1},\ldots,y_{n,r_n})) \to q_0(t_1,\ldots,t_{r_0})$$

thus, by definition of R', we must have:

1. $q = q_0$,
2. $v = t_i[t_{1,[r_1]},\ldots,t_{n,[r_n]}]$,
3. $w = t_j[t_{1,[r_1]},\ldots,t_{n,[r_n]}]$ with,
4. for $k \in [n]$ and $l \in [r_k]$, $t_{k,l} = \langle q_k,l\rangle(x_k)$.

Suppose that both v and w contain an occurrence of $\langle q_k,l\rangle(x_k)$, then both t_i and t_j contain an occurrence of $x_{k,l}$ and since M is linear we have $i = j$ which finaly entails that the two rules are the same, and the occurrences $\langle q_k,l\rangle(x_k)$considered in v and w are in fact a unique occurrence; therefore M is strictly single use with respect to Π_q.

To complete the proof that T' is single use, we now define a collection of *su* mapping of T'.

Given $\sigma \in \Sigma^{(n)}$ and $q_0, \ldots, q_n \in Q$, we define the function $\mathcal{L}_{\sigma,\langle q_1,\ldots,q_n\rangle} : \Pi \times [n] \to \Pi$ to associate Π_{q_i} to (Π_{q_0}, i) if $\sigma(q_1(y_{1,[r_1]}),\ldots,q_n(y_{n,[r_n]})) \to q_0(t_1,\ldots,t_{r_0})$ is in R. The determinism of M trivially implies that $\mathcal{L}_{\sigma,\langle q_1,\ldots,q_n\rangle}$ is actually a function. Now for $\langle\langle q_0,i\rangle(\sigma(x_1,\ldots,x_n)) \to v, q_1,\ldots,q_n\rangle \in R'$, whenever $\langle q_k,l\rangle(x_k)$ occurs in v, by construction, we have that $\langle q_k,l\rangle \in \mathcal{L}_{\sigma,\langle q_1,\ldots,q_n\rangle}(\Pi_{q_0},k) = \Pi_k$. This finally shows that T' is single use (and therefore, as per the remark above, that T realizes a single use transduction).

Bibliography

Bauderon, Michel and Bruno Courcelle (1987). Graph expressions and graph rewritings. *Mathematical Systems Theory*, **20**(2-3):83–127.

Bobaljik, Jonathan David (1999). Adverbs: The hierarchy paradox. *Glot International*, **4**(9-10):27–28.

Chomsky, Noam (1995). *The Minimalist Program*. MIT Press, Cambridge, Massachusetts.

de Groote, Philippe, Glyn F. Morrill, and Christian Retoré, eds. (2001). *Logical Aspects of Computational Linguistics*, volume 2099 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, Berlin.

Engelfriet, Joost (1977). Top-down tree transducers with regular look-ahead. *Mathematical Systems Theory*, **10**:289–303.

Engelfriet, Joost (1994). Graph grammars and tree transducers. In Sophie Tison, ed., *Trees in Algebra and Programming – CAAP'94*, volume 787 of *Lecture Notes in Computer Science*, pp. 15–36. Springer.

Engelfriet, Joost and Linda Heyker (1992). Context-free hypergraph grammars have the same term-generating power as attribute grammars. *Acta Informatica*, **29**(2):161–210.

Engelfriet, Joost and Sebastian Maneth (1999). Macro tree transducers, attribute grammars, and MSO definable tree translations. *Information and Computation*, **154**:34–91.

Engelfriet, Joost and Sebastian Maneth (2000). Tree languages generated by context-free graph grammars. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, eds., *Selected papers from the 6th International Workshop on Theory and Application of Graph Transformations*, volume 1764 of *Lecture Notes in Computer Science*, pp. 15–29. Springer Verlag.

Engelfriet, Joost and Heiko Vogler (1985). Macro tree transducers. *Journal of Computer and System Sciences*, **31**:71–146.

Fülöp, Zoltán, Armin Kühnemann, and Heiko Vogler (2004). A bottom-up characterization of deterministic top-down tree transducers with regular look-ahead. *Information Processing Letters*, **91**:57–67.

Fülöp, Zoltán, Armin Kühnemann, and Heiko Vogler (2005). Linear deterministic multi bottom-up tree transducers. *Theoretical Computer Science*, **347**:276–287.

Harkema, Henk (2001). A characterization of minimalist grammars. In de Groote et al. (2001).

Kobele, Gregory M. (2006). *Generating Copies: An investigation into structural identity in language and grammar*. Ph.D. thesis, University of California, Los Angeles.

Michaelis, Jens (1998). Derivational minimalism is mildly context-sensitive. In M. Moortgat, ed., *Logical Aspects of Computational Linguistics, (LACL '98)*, volume 2014 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, Berlin, Heidelberg, Germany.

Michaelis, Jens (2001). Transforming linear context-free rewriting systems into minimalist grammars. In de Groote et al. (2001).

Michaelis, Jens (2005). An additional observation on strict derivational minimalism. In Gerhard Jäger, Paola Monachesi, Gerald Penn, James Rogers, and Shuly Wintner, eds., *Proceedings of the 10th conference on Formal Grammar and the 9th Meeting on Mathematics of Language*. Edinburgh.

Michaelis, Jens, Uwe Mönnich, and Frank Morawietz (2001). On minimalist attribute grammars and macro tree transducers. In Christian Rohrer, Antje Rossdeutscher, and Hans Kamp, eds., *Linguistic Form and its Computation*, pp. 51–91. CSLI Publications.

Mönnich, Uwe (1997). Adjunction as substitution. In Geert-Jan M. Kruijff, Glyn Morrill, and Richard T. Oehrle, eds., *Formal Grammar 1997: Proceedings of the Conference*, pp. 169–178. Aix-en-Provence.

Morawietz, Frank (2003). *Two-Step Approaches to Natural Language Formalisms*, volume 64 of *Studies in Generative Grammar*. Mouton de Gruyter.

Nesson, Rebecca and Stuart Shieber (2006). Simpler TAG semantics through synchronization. In Paola Monachesi, Gerald Penn, Giorgio Satta, and Shuly Wintner, eds., *Proceedings of the 11th conference on Formal Grammar*, pp. 103–117. CSLI.

Pan, Michael J. (2007). Pomset mcfgs. In *Proceedings of the 10th International Conference on Parsing Technology (IWPT)*.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami (1991). On multiple context-free grammars. *Theoretical Computer Science*, **88**:191–229.

Shieber, Stuart M. (1994). Restricting the weak-generative capacity of synchronous tree-adjoining grammars. *Computational Intelligence*, **10**(4):371–385.

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)*, pp. 377–384. Trento.

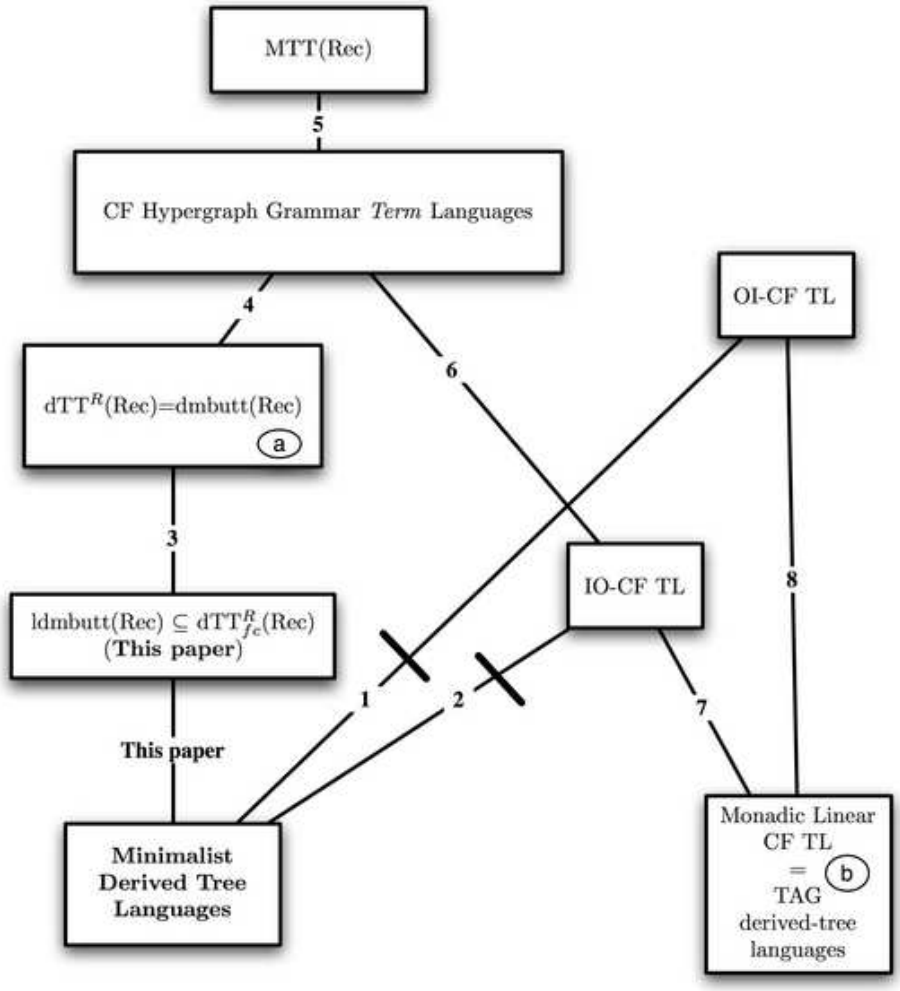
Shieber, Stuart M. and Yves Schabes (1990). Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, volume 3, pp. 253–258. Helsinki,Finland.

Stabler, Edward P. (1997). Derivational minimalism. In Christian Retoré, ed., *Logical Aspects of Computational Linguistics*, volume 1328 of *Lecture Notes in Computer Science*, pp. 68–95. Springer-Verlag, Berlin.

Stabler, Edward P. and Edward L. Keenan (2003). Structural similarity within and among languages. *Theoretical Computer Science*, **293**:345–363.

Vijay-Shanker, K., David Weir, and Aravind Joshi (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Meeting of the Association for Computational Linguistics*, pp. 104–111.

Weir, David J. (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.



TL Tree Languages **Rec** Regular Tree Languages **CF** Context Free **OI** Outside-In **IO** Inside Out **TAG** Tree Adjoining Grammars	**MTT** Macro Tree Transducer **dTT$^R_{fc}$** Deterministic Top Down Tree Transducer with Regular Look-Ahead (Finite Copy) — see this paper. **(l)dmbutt** (Linear) Deterministic Multi-Bottom-Up Tree Transducer — see this paper.	**a** was proved in Fülöp et al. (2004) **b** was proved in Mönnich (1997) **1, 2** these non inclusions were proved in Michaelis (2005) by looking at the string languages **4,5,6** are presented in Engelfriet and Heyker (1992) **7,8** are obvious.

Figure 9.2: Minimalist derived trees and friends