wMSO Theories as Grammar Formalisms

James Rogers [1]

Computer Science Department, Earlham College, Richmond, Indiana, USA

Abstract

We explore the use of weak monadic second-order languages over structures of varying dimension as specification languages for grammars and automata, focusing, in particular, on the extension of the longstanding results characterizing the regular and context-free languages in terms of definability in wS1S (one-dimensional) and wSnS (two-dimensional), respectively, to a characterization of the Tree-Adjoining Languages in terms of definability in the weak monadic second-order theory of certain three-dimensional tree-like structures. We then explore the application of these results to aspects of an existing large-scale Tree-Adjoining Grammar for English and close with some speculation on the feasibility of this approach as a means of building and maintaining such grammars.

Key words: Model-theoretic syntax, wSnS, Descriptive complexity, Tree-adjoining grammar
1991 MSC: 68Q45, 68Q42, 68Q19

1 Introduction

Much of the research in applying logical apparatus to syntax has had a distinct proof-theoretic flavor. For the most part, use of model-theoretic tools has focused on providing semantics for grammar formalisms. In this paper we explore a sequence of model-theoretic results stretching from 1960 to the present establishing descriptive characterizations of standard grammar- and automata-theoretic complexity classes. The associated logical languages provide a means of defining languages within the various classes in a fully declarative way, abstracting away from the details of the underlying generative mech-

---

Email address: jrogers@cs.earlham.edu (James Rogers).

anisms. These results have application in theoretical explorations of the complexity of linguistic constraints and in investigations of the linguistic content of existing grammars, and, because they provide a common framework for formalization, they provide a flexible mechanism for comparing theories expressed within disparate grammatical systems. Moreover, because they include effective translations of formulae into automata—albeit not without significant feasibility issues, a consequence of the extraordinary conciseness of the logical languages—they have potential for application in practice as a means of specifying grammars directly in terms of the linguistic principles on which they are based.

This work has, at its root, results of Büchi [1] and of Elgot [2] dating to the late '50's which establish decidability of the monadic second-order theory of the natural numbers with successor (what we would now call S1S) by automata-theoretic techniques. These were extended, in the late '60's, by Doner [3] and by Thatcher and Wright [4] (for the weak fragment) and Rabin [5] (the full MSO theories) to the structure of multiple successors—to trees (SnS). From a computational perspective, what is most attractive about these results is that the proofs employ constructions that, given an MSO formula, produce a finite-state automaton that accepts exactly the set of structures that satisfy it. This has been exploited in verification of the temporal properties of software and hardware systems by encoding both the behavior of the system, as realized, and its required behavior in an MSO formula that defines the class of realized behaviors which fail to satisfy the required behavior, translating this into an automaton, and then using automata-theoretic techniques to either prove that the class is empty or to characterize the set of behaviors which fail [6–8].

If we restrict our attention to finite models, the constructions produce ordinary finite-state automata over strings or trees and the weak MSO fragment suffices. Under these circumstances, definability in wS1S characterizes the regular string languages and definability in wSnS the recognizable sets of labeled trees (which yield exactly the Context-Free string languages). Thus any theory of syntax that is based on relationships within strings or within trees that can be expressed in the wMSO theory of these structures licenses a regular or, respectively, Context-Free language. We have employed this characterization to establish that a substantial fragment of a standard GB account of English syntax licenses a CFL [9], to provide fully declarative static accounts of superficially dynamic aspects of GPSG [10], and to explore the distinctions between these approaches [11]. The construction, itself, has been employed to produce recognizers for aspects of the GB account [12,13].

The fundamental limitation of these results is the weakness of the language classes they characterize—to capture the range of natural languages similar results for larger classes are necessary. Recently, by viewing the step from strings to trees as a step from one- to two-dimensional structures and then

d	Local	Composite
0	.	(none)
1	.	

Fig. 1. Some 0- and 1-dimensional trees.

generalizing this to tree-like structures of arbitrary dimension, we have extended these results to an infinite hierarchy of languages that coincides with Weir's version of the Control Language Hierarchy [14]. What is particularly attractive about this approach is the fact that it provides a uniform notion of grammars and automata which permits essentially standard proofs of the properties of the local and recognizable sets, as well as the translation from wMSO formulae to automata, to be applied at all levels of the hierarchy—the dimension becomes a parameter of the constructions that determines the type of structures it manipulates but which has no substantive role in the proof itself.

From the linguistic perspective, what makes this hierarchy attractive is the fact that the finite-state automata over the three-dimensional structures are, in essence, Tree-Adjoining Grammars—the foundation of a considerable amount of current work in applied computational linguistics. Again, there are a number of theoretical issues that can be clarified by the model-theoretic perspective. But, these results may prove to be most useful in addressing the overwhelming complexity of building and maintaining these large grammars. To a large extent, this complexity is an artifact of the need to distribute the effect of linguistic constraints throughout very large sets of elementary trees. By defining these constraints with wMSO formulae this process can be left to the automaton construction algorithm. In effect, the wMSO formulae become a sort of logic-programming for TAGs with the automaton construction serving as a compiler. This goal is not without significant difficulty, though. The wMSO formulae are extraordinarily concise—the asymptotic rate of growth of the automata wrt the size of the formulae is not even elementary-recursive. As with symbolic model-checking the crucial issue in realizing this program is avoiding the infeasible automata.

In the next two sections we introduce the hierarchy of relational structures and corresponding weak monadic second-order theories that form the foundation of these results. We then (Section 4) introduce the grammars and automata over these structures and provide the connection between definability in the MSO theories and these more traditional generative frameworks for theories of syntax (Sections 5 and 6). The remainder of the paper looks at particular applications to Tree-Adjoining Grammars, first sketching the equivalence of
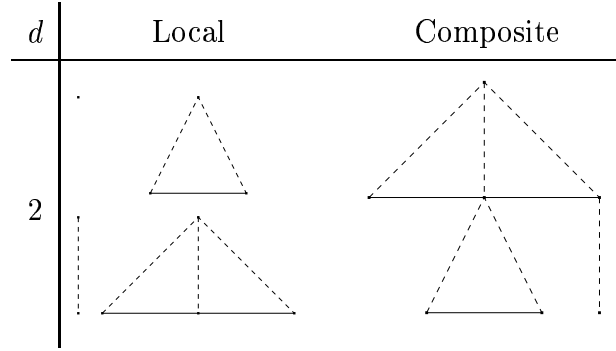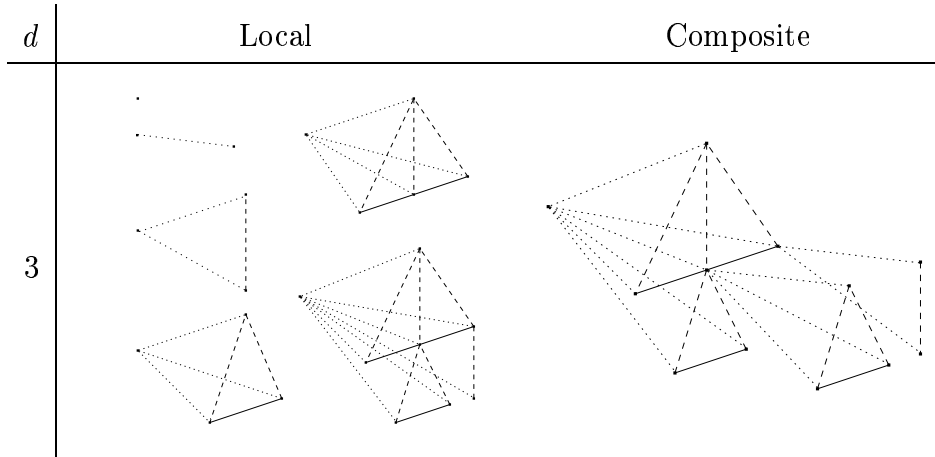
Fig. 2. Some 2-dimensional trees.

Fig. 3. Some 3-dimensional trees.

definability at the three-dimensional level and the strong generative capacity of TAG (Section 7), then looking at the treatment of aspects of the current XTAG grammar for English within this logical framework (Section 8), and ending with some speculation about the potential of this framework for simplifying TAG grammar development and maintenance.

2   Multi-Dimensional Tree Domains

We begin by developing a hierarchy of classes of multi-dimensional structures. At the dth level these are assemblages of d-dimensional *local structures*— structures of depth at most 1 in their major (d^{th}) dimension (see Figures 1, 2, and 3). These local structures consist of a point (the *root* of the local structure) and the set of its successors in the dth dimension (the *yield* of local structure) which is required to be an (arbitrary) $(i-1)$-dimensional structure. We also admit both *empty* local structures and *trivial* local structure (in which the yield is empty). Thus, even trivial local i-dimensional structures have an $(i-1)$-dimensional structure as their yield, albeit an empty one.
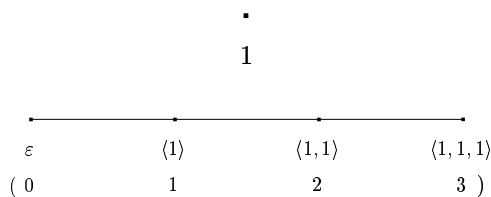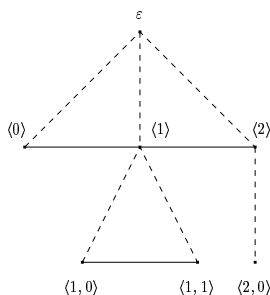
4

Fig. 4. Point and string domains.

Fig. 5. A 2-dimensional tree domain.

Composite structures are constructed by identifying the root of one local structure with a point in the yield of another. Each component structure has a unique root, which is a member of only a single local structure. Every other point is a member of exactly two local structures—it is a point in the yield of one and the root of another (which is, of course, possibly maximal). These structures are all tree-like in the sense that every point is reachable from the root by exactly one path of major dimension successor relations. For concreteness, we can assume a canonical form for their domains in which each point is represented by its *address*—an encoding of the path leading to it.

At the base of the hierarchy (which we will refer to as the 0th level) we have *point domains*: every point is maximal and each composite structure contains a single local structure. Hence each structure has a domain consisting of a single point.

At the first level we have 1-dimensional structures—the *string domains* (Figure 4). Here each (non-maximal) point has a (single) point successor. The yields of local 1-dimensional structures have cardinality less than or equal to one. In essence, they are pairs of points. In following paths from the root, at each point there is at most a single successor from which to choose. If we represent that choice as '1' then *string addresses* are sequences of '1's in which the length of the address of a point is just its depth. Then the canonical representation of a composite domain is a prefix closed set of sequences of '1's, with the root at address ε. If we interpret these sequences of '1's as unary numerals we get that the canonical domains of the composite structures are initial segments of $\mathbb{N}$ with the root at address '0'. These are the structures studied by Büchi and by Elgot.
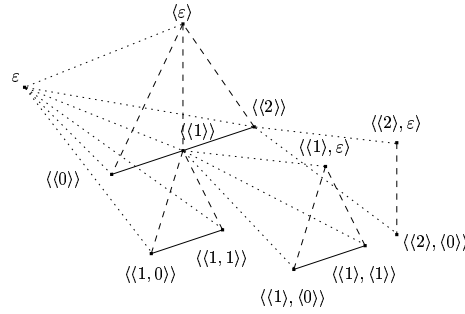
5

Fig. 6. A 3-dimensional tree domain.

The second level is the class of *tree domains* of Gorn [15] (Figure 5). Here each point has a set of successors (in the 2nd dimension) which form a (possibly empty) string domain (ordered in the 1st dimension). Hence, the local structures are depth-1 ordered trees: a root and its set of children (as a sequence of points). Now at each point in a path from the root of a composite structure one has a choice of any of the successors of that point. We will represent that choice by its string address in the yield of the local structure. Thus, *tree addresses* will be sequences of string addresses: second-order sequences of '1's, and canonical tree domains will be sets of these second-order sequences which are *hereditarily prefix closed* in the sense that they are prefix closed wrt the top level sequences and, for every second-order sequence s, the set of first-order sequences w such that $s \cdot \langle w \rangle$ is in the set is also prefix closed. [2] At this two-dimensional level, the notion of hereditary prefix closure simply requires the addresses of the children of a point in the set to be a properly formed (prefix closed) string domain. If we understand the ith element of the string of children of a point to be its ith successor we obtain the structures studied by Doner, by Thatcher and Wright, and (admitting infinite sequences of children) by Rabin.

This process iterates. The local structures at level $d+1$ are formed by adding a $(d+1)^{\text{st}}$-dimensional root to an arbitrary composite d-dimensional structure, the domain of which forms the set of its successors in the $(d+1)^{\text{st}}$ dimension and the canonical domains of the composite $(d+1)^{\text{st}}$-dimensional structures are hereditarily prefix closed $(d+1)^{\text{st}}$-order sequences of '1's. At the third level the local structures are pyramidal with the root at the apex and the tree of its successors forming the base (Figure 6). For convenience, we will refer to the set of all d^{th}-order sequences of '1's as: $^{d}1$.

---

[2] Here '·' represents concatenation, which we will always take to be an operation on sequences of the same order. In general, we will use s, t, etc. and w, v, etc. in this way, with s denoting sequences of the next higher order than w. We will also, occasionally, use p, q, etc. to denote sequences of the next higher order yet.

Adopting the terminology of the second level, we will refer to a structure in the dth-dimensional level of this hierarchy as a *d-dimensional tree domain*, denoted Td, with the set of all Td being denoted $\mathbb{T}^d$. The *depth* of a Td (in its major dimension) is the length of the longest sequence it includes (just the length of the top level sequence, independent of the length of the sequences it may contain). We will refer to the set of yields of the local structures comprising a Td as the set of its $((d-1)$-dimensional) *child structures*. We will refer to the set of all child structures (in any dimension) occurring in a Td as its set of *component structures*.

The *branching factor* of a Td *at a given dimension* is one plus the maximum depth of the component structures it contains in that dimension. The (overall) *branching factor* of a Td is the maximum of its branching factors at all dimensions strictly less than d. In a T3, for example, the branching factor is one plus the larger of the maximum depth of the trees it contains and the maximum length of the strings it contains. A Td is *n-branching* iff its branching factor is no greater than n.

For any alphabet Σ, a Σ-*labeled Td* is a pair $\langle T, \tau \rangle$ where T is a Td and $\tau : T \to \Sigma$ is an assignment of labels in Σ to the nodes in T. We will denote the set of all Σ-labeled Td as $\mathbb{T}^d_\Sigma$. We will denote the set of all Σ-labeled, n-branching, Td as $\mathbb{T}^{n,d}_\Sigma$.

3 wSnTd

We are now in a position to build relational structures on d-dimensional tree domains. Let T^d_n be the *complete n-branching Td*—that in which every point has a child structure that has depth n in all its $(d-1)$ dimensions. Let

$$\mathsf{T}^d_n \stackrel{\text{def}}{=} \langle T^d_n, \lhd_i \rangle_{1 \leq i \leq d}$$

where, for all $x, y \in T^d_n$:

$$x \lhd_d y \stackrel{\text{def}}{\Longleftrightarrow} y = x \cdot \langle s \rangle$$
$$x \lhd_{d-1} y \stackrel{\text{def}}{\Longleftrightarrow} x = p \cdot \langle s \rangle \text{ and } y = p \cdot \langle s \cdot \langle w \rangle \rangle$$
$$\vdots$$
$$x \lhd_1 y \stackrel{\text{def}}{\Longleftrightarrow} x = p \cdot \langle s \cdot \langle \cdots \langle w \rangle \cdots \rangle \rangle \text{ and } y = p \cdot \langle s \cdot \langle \cdots \langle w \cdot 1 \rangle \cdots \rangle \rangle$$

(which is to say that $x \lhd_i y$ iff x is the immediate predecessor of y in the ith-dimension).

7

The *weak monadic second-order language* of T_n^d includes constants for each of
the relations (we let them stand for themselves), the usual logical connectives,
quantifiers and grouping symbols, and two countably infinite sets of variables,
one ranging over individuals (for which we employ lowercase) and one ranging
over *finite* subsets (for which we employ uppercase). If $\varphi(x_1, \ldots, x_n, X_1, \ldots, X_m)$
is a formula of this language with free variables among the x_i and X_j, then
we will assert that it is satisfied in T_n^d by an assignment $\boldsymbol{s}$ (mapping the 'x_i's
to individuals and 'X_j's to finite subsets) with the notation: $\mathsf{T}_n^d \models \varphi\,[\boldsymbol{s}]$.

The *weak monadic second-order theory of* T_n^d, denoted wSnTd, is the set of
all sentences of this language that are satisfied by T_n^d. (wS1T1 is equivalent
to wS1S in the sense of interinterpretability, as is wS1Td for all d. wSnT2 is
interinterpretable with wSnS for all $n \geq 2$.)

A set $\mathbb{T}$ of Σ-labeled Td is definable in wSnTd iff there is a formula $\varphi_{\mathbb{T}}$ with
free variables among X_T (interpreted as the domain of a tree) and X_σ for each
$\sigma \in \Sigma$ (interpreted as the set of σ-labeled points in T), such that

$$\langle T, \tau \rangle \in \mathbb{T} \iff \mathsf{T}_n^d \models \varphi_{\mathbb{T}}\,[X_T \mapsto T, X_\sigma \mapsto \{p \mid \tau(p) = \sigma\}].$$

4 Td Grammars and Automata

Mimicking the development of multi-dimensional tree domains, we can define
automata over labeled Td as a generalization of automata over labeled tree
domains which, in turn, can be understood as an analogous generalization
of ordinary finite-state automata over strings (labeled string domains). A Td
automaton with state set Q and alphabet Σ is a finite set:

$$\mathcal{A}^d \subseteq \Sigma \times Q \times \mathbb{T}_Q^{d-1}.$$

The interpretation of a tuple $\langle \sigma, q, \mathcal{T} \rangle \in \mathcal{A}^d$ is that if a node of a Td is
labeled σ and $\mathcal{T}$ encodes the assignment of states to its children, then that
node may be assigned state q. This is a "bottom-up" interpretation. There
is an analogous "top-down" interpretation, but for all $d \geq 2$ automata that
are deterministic under the top-down interpretation are strictly weaker than
those that are non-deterministic, while those that are deterministic under the
bottom-up interpretation are equivalent to the non-deterministic variety. It
should be emphasized that the only place the distinction between top-down
and bottom-up arises is in the definition of determinism. These automata are
interpreted purely declaratively, as licensing assignments of states to nodes.

A *run* of a Td automaton $\mathcal{A}$ on a Σ-labeled Td $\mathcal{T} = \langle T, \tau \rangle$ is an assignment
$r : T \to Q$ of states in Q to nodes in T in which each assignment is licensed by

$\mathcal{A}$. Note that this implies that a maximal node (wrt to the major dimension) labeled σ may be assigned state q only if there is a tuple $\langle \sigma, q, \emptyset \rangle \in \mathcal{A}^d$ where $\emptyset$ is the empty $\mathrm{T}(d-1)$. If we let $Q_0 \subseteq Q$ be any set of *accepting states*, then the set of (finite) Σ-labeled $\mathrm{T}d$ recognized by $\mathcal{A}$, relative to Q_0, (denoted $\mathcal{A}(Q_0)$) is that set for which there is a run of $\mathcal{A}$ that assigns the root a state in Q_0.

Let $\mathrm{Ch}(T, s) \stackrel{\mathrm{def}}{=} \{w \in {}^{(d-1)}1 \mid s \cdot \langle w \rangle \in T\}$ and

$$\langle T, r \rangle \, |_{\mathrm{Ch}(T, s)} \stackrel{\mathrm{def}}{=} \quad \langle \mathrm{Ch}(T, s), \{w \mapsto r(s \cdot \langle w \rangle) \mid w \in \mathrm{Ch}(T, s)\} \rangle \,.$$

That is, $\mathrm{Ch}(T, s)$ is the $\mathrm{T}(d-1)$ yield of the local $\mathrm{T}d$ rooted at s in T and $\langle T, r \rangle \, |_{\mathrm{Ch}(T, s)}$ the corresponding Σ-labeled $\mathrm{T}(d-1)$.

Then

$$\mathcal{A}(Q_0) \stackrel{\mathrm{def}}{=} \{\mathcal{T} = \langle T, \tau \rangle \mid T \text{ finite} \text{ and there exists } r : T \to Q \text{ such that}$$
$$r(\varepsilon) \in Q_0 \text{ and for all } s \in T, \left\langle \tau(s), r(s), \langle T, r \rangle \, |_{\mathrm{Ch}(T, s)} \right\rangle \in \mathcal{A}\}.$$

Definition 1 *A set of Σ-labeled $\mathrm{T}d$ is* recognizable *iff it is $\mathcal{A}(Q_0)$ for some $\mathrm{T}d$ automaton $\mathcal{A}$ and set of accepting states Q_0.*

Similarly, a $\mathrm{T}d$ grammar over Σ is a finite set:

$$\mathcal{G}^d \subseteq \Sigma \times \mathbb{T}_\Sigma^{d-1}.$$

and

$$\mathcal{G}(\Sigma_0) \stackrel{\mathrm{def}}{=} \{\mathcal{T} = \langle T, \tau \rangle \mid T \text{ finite}, \tau(\varepsilon) \in \Sigma_0, \text{ and}$$
$$\text{for all } s \in T, \langle \tau(s), \langle T, \tau \rangle \, |_{\mathrm{Ch}(T, s)} \rangle \in \mathcal{G}\}.$$

Note that T2 grammars are, in essence, CFGs viewed as sets of local trees rather than sets of productions, generalized slightly in that there may be multiple start symbols and the terminals (those members of Σ licensed to label nodes maximal in the major dimension) may be rewritten.

Definition 2 *A set of Σ-labeled $\mathrm{T}d$ is* local *iff it is $\mathcal{G}(\Sigma_0)$ for some $\mathrm{T}d$ grammar $\mathcal{G}$ and set of start symbols Σ_0.*

As is well known from the 1- and 2-dimensional levels [16,17], the recognizable sets of $\mathrm{T}d$ are exactly the projections of the local sets of $\mathrm{T}d$. That is, if $\mathbb{T}$, a set of Σ-labeled $\mathrm{T}d$, is $\mathcal{A}(Q_0)$ then it is $\pi_1(\mathcal{G}(\Sigma \times Q_0))$ where $\mathcal{G}$ is the $\mathrm{T}d$ grammar over $\Sigma \times Q$ in which the states of the runs of $\mathcal{A}$ explicitly label the $\mathrm{T}d$.

Lemma 3 ([16,17]) *A set of Σ-labeled Td is recognizable iff it is a projection of a local set of Γ-labeled Td, for some Γ.*

4.1 Uniform Properties of Recognizable Sets

The strength of the uniform definition of Td automata is that many, even most, properties of the sets they recognize can be proved uniformly—independently of their dimension. For instance, a Td automaton is *deterministic* with respect to a branching factor n (in the bottom-up sense) iff

$$(\forall \sigma \in \Sigma, \mathcal{T} \in \mathbb{T}_Q^{n,d-1})(\exists! q \in Q)[\langle \sigma, q, \mathcal{T} \rangle \in \mathcal{A}].$$

(The quantifier $\exists!$ should be read "exists exactly one".)

It is easy to show, using a standard powerset-construction, that (bottom-up) determinism does not affect the recognizing power of Td automata of any dimension. Given $\mathcal{A} \subseteq \Sigma \times Q \times \mathbb{T}_Q^{n,d-1}$, let

$$\hat{\mathcal{A}} \subseteq \Sigma \times \boldsymbol{\mathcal{P}}(Q) \times \mathbb{T}_{\boldsymbol{\mathcal{P}}(Q)}^{n,d-1} =$$
$$\{\langle \sigma, Q_1, \langle T, \tau' \rangle \rangle \mid Q_1 \subseteq Q, \ \tau' : T \to \boldsymbol{\mathcal{P}}(Q),$$
$$q \in Q_1 \Leftrightarrow (\exists \tau : T \to Q)[\langle \sigma, q, \langle T, \tau \rangle \rangle \in \mathcal{A} \wedge (\forall x \in T)[\tau(x) \in \tau'(x)] \,]\}$$

and

$$\hat{Q}_0 \stackrel{\mathrm{def}}{=} \{Q_i \subseteq Q \mid Q_i \cap Q_0 \neq \emptyset\}.$$

It is easy to verify that $\hat{\mathcal{A}}$ is deterministic and that $\hat{\mathcal{A}}(\hat{Q}_0) = \mathcal{A}(Q_0)$. More importantly, while the dimension of the Td automaton parameterizes the type of the objects manipulated by the proof, it has no effect on the way in which they are manipulated—the proof itself is essentially independent of the dimension.

Similar uniform proofs can be obtained for closure of the class of recognizable sets under projection, cylindrification, and Boolean operations and for decidability of emptiness.

4.2 A Myhill-Nerode Characterization

Theorem 4 *Suppose $\mathbb{T} \subseteq \mathbb{T}_\Sigma^d$. For all $\mathcal{T}_1, \mathcal{T}_2 \in \mathbb{T}_\Sigma^d$, let $\mathcal{T}_1 \equiv_\mathbb{T} \mathcal{T}_2$ iff, for every tree $\mathcal{T} \in \mathbb{T}_\Sigma^d$ and point s in the domain of $\mathcal{T}$, the result of substituting $\mathcal{T}_1$ at s*

10

in $\mathcal{T}$ is in $\mathbb{T}$ iff the result of substituting $\mathcal{T}_2$ is. Then $\mathbb{T}$ is recognizable iff $\equiv_{\mathbb{T}}$ has finite index.

In the one-dimensional case $\equiv_{\mathbb{T}}$ is the Nerode equivalence for the reversed language. The characterization, in the two dimensional case, is well known as well [18]. Again, the proof at all levels is a simple lift of the proof at the one-dimensional level: in one direction it follows nearly immediately from the finiteness of the set of states, in the other it is based on a construction of an automaton using $\mathbb{T}^d_{\Sigma}/_{\equiv_{\mathbb{T}}}$ as the set of states.

5 Definability and Recognizability

It should be reasonably clear that the recognizable sets of Σ-labeled Td are definable in wSnTd. One encodes each production of the automaton recognizing the set as a formula in the language of wSnTd, with free variables for Σ and Q, which will be satisfied at the root of a local tree by an assignment s iff the assignment labels it (in the same sense as in the definition of definability) consistently with that production. One then combines these into a formula requiring every point in the domain of the tree to satisfy one of these formulae. Finally, one hides the states by existentially binding them.

The proof that every wSnTd definable set of Σ-labeled Td (for finite n) is recognizable is, yet again, a direct lift of the corresponding proofs at the 1- and 2-dimensional levels [1–4]. One first reduces to a language in which only set variables occur and the only predicates are subset and predicates for the relations between singleton subsets corresponding to immediate domination in the various dimensions. Then one defines automata for each of these predicates. The extension to arbitrary formulae is obtained from the constructions of the proofs that the class of recognizable sets is closed under Boolean operations and (for existential quantification) projection.

Theorem 5 *A set of Σ-labeled Td is recognizable iff it is definable in wSnTd, for some $n < \omega$.*

From a practical perspective, the most attractive aspect of this result is the fact that the proof is constructive: it provides effective means of translating arbitrary sets of formulae in the language of wSnTd with free variables in Σ into Td automata that license exactly the set of Σ-labeled Td that satisfy the formulae. Hence, we can define sets of Σ-labeled Td in terms of highly abstract logical constraints and employ the construction of the proof to translate these into automata that can be processed relatively efficiently.

The weakness in this plan is the extraordinary conciseness of the logical for-

11

mulae in terms of the size of the corresponding automata. As Meyer [19] has
pointed out, the size of the automata generated by the construction is non-
elementary in the size of the formulae—the number of states is proportional
to a constant raised to a stack of exponents the height of which is proportional
to the number of quantifier alternations in the formula (a constant tetrated
to the number of alternations). Consequently, it is easy to describe sets which
are infeasible in the sense that the automata that recognize them are of im-
practical size (and take an impractically long time to build) regardless of their
dimension. This has been one of the main limiting factors in application of
these results at any level and the problem of how to identify classes of formu-
lae that are impractical to implement is very much still an open issue. We will
have more to say about this in the final section of the paper.

6 Yields of Higher Dimensional Structures

So far, our Td grammars and automata define only sets of Σ-labeled Td. We
obtain a string language from such a set by applying a *yield* operation $d - 1$
times. Given $\mathcal{T}$, a Σ-labeled Td we would like the *yield* of $\mathcal{T}$ to be the T$(d-1)$
obtained by restricting $\mathcal{T}$ to its maximal points with respect to $\lhd_d$. For the T2
case this is straightforward, but for the higher-dimensional cases we need to
specify the way in which the yield of the structure dominated by a node in the
d^{th} dimension splices together with the yield of that node's children in the
$(d-1)^{\text{st}}$ dimension. In Figure 6, for instance, the issue is whether $\langle\langle 1, 0 \rangle\rangle$ and
$\langle\langle 1, 1 \rangle\rangle$ should be the children of $\langle\langle 1 \rangle, \langle 0 \rangle\rangle$ or $\langle\langle 1 \rangle, \langle 1 \rangle\rangle$ in the two-dimensional
yield.

Our approach is to distinguish a unique node in the frontier of the yield of
each $(d-1)$-dimensional component, its *foot*, which will serve as the "splicing
point" for the yields. We distinguish the foot by extending our structures with
d distinguished subsets $H_1, \ldots, H_d$. Each H_i is required to pick out exactly
one child of each non-empty local Ti in the structure—the *head* of that Ti.
Then, in each Td, there will be a unique sequence of points in H_d forming a
path from the root to a node on the frontier. We will refer to this path as the
(principle) spine of the structure. The foot will be the maximal point of the
spine.

6.1 Σ-Labeled Headed Td

Given a suitable selection of such H_i we will be able to extend each $\lhd_i$ to a
domination relation $\lhd_i^+$ that is preserved under the yield operation. We can
then adopt, as our models, *headed Td*—relational structures based on $\lhd_i^+$ rather

than $\lhd_i$—in which case the $(d-1)$-dimensional yield of a Td $\mathcal{T}$ will just be the reduct to $\lhd_i^+$, $i < d$ of the restriction of $\mathcal{T}$ to its maximal elements wrt $\lhd_d^+$.

Definition 6 *A Σ-Labeled Headed Td is a structure:*

$$\mathcal{T} = \langle T, \lhd_i^+, R_i, H_i, P_\sigma \rangle_{1 \leq i \leq d, \sigma \in \Sigma},$$

where the components are defined in the remainder of this section.

Our goal in this section is two-fold: first to define the class of structures we have in mind and, second, to demonstrate that they form a wMSO definable class of ordinary Σ-labeled Td. Towards the first goal, we define the intended interpretation of the predicates $\lhd_i^+$, R_i, and H_i, $1 \leq i \leq d$. Towards the second, we show how each aspect of these definitions can be expressed as wMSO formulae over the signature of T_n^d (employing the the defined predicates as they become available). The result is a set of axioms which can be adjoined to the definition of a set of Σ-labeled headed Td transforming it into a definition of a set of $\Sigma \cup \{H_i, R_i \mid 1 \leq i \leq d\}$-labeled ordinary T$d$.

The domain, T, is a d-dimensional tree domain: $T \in \mathbb{T}^d$.

The R_i are the sets of roots of i-dimensional constituent structures of $\mathcal{T}$—the minimal points wrt $\lhd_i$. There is a single root in R_d and one root in R_i for each i-dimensional component structure. In Figure 6 $R_3 = \{\varepsilon\}$, $R_2 = \{\langle\varepsilon\rangle, \langle\langle 1\rangle, \varepsilon\rangle, \langle\langle 2\rangle, \varepsilon\rangle\}$ and $R_1 = \{\langle\langle 0\rangle\rangle, \langle\langle 1, 0\rangle\rangle, \langle\langle 1\rangle, \langle 0\rangle\rangle, \langle\langle 2\rangle, \langle 0\rangle\rangle\}$. In general, R_i is the set of all addresses that end in an empty i^{th}-order sequence:

$$x \in R_d \stackrel{\text{def}}{\Longleftrightarrow} x = \varepsilon$$

$$x \in R_{d-1} \stackrel{\text{def}}{\Longleftrightarrow} x = p \cdot \langle\varepsilon\rangle, \quad p \text{ a } d^{\text{th}}\text{-order sequence },$$

$$\vdots$$

$$x \in R_1 \stackrel{\text{def}}{\Longleftrightarrow} \begin{cases} x = \varepsilon & \text{if } d = 1, \\ x = p \cdot \langle s \cdot \langle \cdots w \cdot \langle\varepsilon\rangle \cdots \rangle\rangle, \\ \qquad w \text{ a } 2^{\text{nd}}\text{-order sequence, otherwise.} \end{cases}$$

As with the other components of our structures, we employ R_i both as a monadic predicate and to denote its interpretation in the structure. The intended interpretation of the R_i is, of course, wMSO definable:

$$R_i(x) \leftrightarrow (\forall y)[\neg y \lhd_i x].$$

Hence, in translating definitions of sets of Σ-labeled headed Td into definitions of sets of ordinary labeled Td we can take the R_i to be existentially bound

13

second-order variables with their satisfying assignments restricted by adjoining these formulae to the definition.

In defining the intended interpretations of the H_i and $\lhd_i^+$ we will employ auxiliary relations $\bar{\lhd}_i$ denoting proper domination in the ith -dimension (the irreflexive transitive closure of $\lhd_i$) within an i-dimensional component:

$$x \ \bar{\lhd}_d \ y \overset{\text{def}}{\Longleftrightarrow} y = x \cdot p, \ p \neq \varepsilon$$
$$x \ \bar{\lhd}_{d-1} \ y \overset{\text{def}}{\Longleftrightarrow} x = p \cdot \langle s \rangle \text{ and } y = p \cdot \langle s \cdot t \rangle, t \neq \varepsilon$$
$$\vdots$$
$$x \ \bar{\lhd}_1 \ y \overset{\text{def}}{\Longleftrightarrow} x = p \cdot \langle s \cdot \langle \cdots \langle w \rangle \cdots \rangle \rangle \text{ and}$$
$$y = p \cdot \langle s \cdot \langle \cdots \langle w \cdot v \rangle \cdots \rangle \rangle, v \in 1^+.$$

Note that these relations are wMSO definable by the formulae:

$$x \ \bar{\lhd}_i \ y \equiv (\forall X)[(X(y) \wedge (\forall z_1, z_2)[(X(z_1) \wedge z_2 \lhd_i z_1) \to X(z_2)]) \to X(x)].$$

Which requires every subset of the domain that both includes y and is downward (i.e., "root"-ward) closed wrt $\lhd_i$ to include x as well. Hence, we may employ the $\bar{\lhd}_i$ without extending the descriptive power of wSnTd, simply by taking the relations to be syntactic shorthand for the corresponding definition.[3]

The points in the H_i are required to be distributed such that there is exactly one member of H_i in the yield of each non-empty local i-dimensional structure.

$$(\forall t)[(\exists s)[t \lhd_i s] \Rightarrow (\exists! s)[H_i(s) \wedge t \lhd_i s]].$$

Referring, again, to Figure 6: H_3 must include exactly one point from each of the sets $\{\langle \varepsilon \rangle, \langle \langle 0 \rangle \rangle, \langle \langle 1 \rangle \rangle, \langle \langle 2 \rangle \rangle, \langle \langle 1, 0 \rangle \rangle, \langle \langle 1, 1 \rangle \rangle\}$, $\{\langle \langle 1 \rangle, \varepsilon \rangle, \langle \langle 1 \rangle, \langle 0 \rangle \rangle, \langle \langle 1 \rangle, \langle 1 \rangle \rangle\}$ and $\{\langle \langle 2 \rangle, \varepsilon \rangle, \langle \langle 2 \rangle, \langle 0 \rangle \rangle\}$; H_2 must include exactly one point from each of the sets $\{\langle \langle 0 \rangle \rangle, \langle \langle 1 \rangle \rangle, \langle \langle 2 \rangle \rangle\}$, $\{\langle \langle 1, 0 \rangle \rangle, \langle \langle 1, 1 \rangle \rangle\}$, $\{\langle \langle 1 \rangle, \langle 0 \rangle \rangle, \langle \langle 1 \rangle, \langle 1 \rangle \rangle\}$ and from $\{\langle \langle 2 \rangle, \langle 0 \rangle \rangle\}$; and H_1 must include exactly one point from each of the sets $\{\langle \langle 1 \rangle \rangle\}$, $\{\langle \langle 2 \rangle \rangle\}$, $\{\langle \langle 1, 1 \rangle \rangle\}$, and $\{\langle \langle 1 \rangle, \langle 1 \rangle \rangle\}$.

---

[3] We will generally distinguish explicit definitions of this sort by using '$\equiv$'. All new non-monadic predicates must be explicitly defined in this way. In contrast, new monadic first-order predicates may be implicitly or even inductively defined (as we do here) since all such predicates are MSO definable. More precise definitions of these varieties of definability along with proofs of their relationships with MSO definability can be found in [9].

It is worth noting that H_i is disjoint from all R_j, $j \geq i$ since the points in R_j are not in the yield of any local Ti. On the other hand, H_i may well include points in R_j for $j < i$ and, in fact, we will shortly further restrict the interpretation of the H_i to require the i-dimensional heads to be chosen from among the j-dimensional roots and particular subsets of the j-dimensional heads for $j = i - 1$. Note, also, that there is no freedom in the interpretation of H_1—there is but a single point in the yield of any local T1.

By requiring each non-trivial local Ti to have a single point in its yield distinguished by inclusion in H_i we ensure that there is a unique sequence of points in H_i, ordered by $\lhd_i$, leading from a given point to a leaf of the i-dimensional component containing it. We will refer to such a sequence as a *spine*. The set of elements of the spine extending from a point t in $\mathcal{T}$ is:

$$\mathrm{SP}_i(t, \mathcal{T}) \stackrel{\mathrm{def}}{=} \{t\} \cup$$
$$\{s \in H_i \mid t \mathrel{\bar{\lhd}_i} s \text{ and } (\forall s')[(t \mathrel{\bar{\lhd}_i} s' \text{ and } s' \mathrel{\bar{\lhd}_i} s) \Rightarrow s' \in H_i]\}.$$

The *principle* spine of an i-dimensional component structure is the spine starting at its root. The auxiliary predicate PSP_i picks out points on some i-dimensional principle spine:

$$\mathrm{PSP}_i(x) \stackrel{\mathrm{def}}{\Longleftrightarrow} (\exists y \in R_i)[x \in \mathrm{SP}_i(y, \mathcal{T})].$$

Again, this is wMSO definable:

$$\mathrm{PSP}_i(x) \leftrightarrow (R_i(x) \vee$$
$$H_i(x) \wedge (\exists y)[R_i(y) \wedge y \mathrel{\bar{\lhd}_i} x \wedge (\forall z)[(y \mathrel{\bar{\lhd}_i} z \wedge z \mathrel{\bar{\lhd}_i} x) \rightarrow H_i(z)]]).$$

The restrictions that we have placed on the interpretation of the H_i so far suffice for their primary purpose—to pick out principle spines. For technical reasons (having to do with persistence of heads under various restrictions of the structures) we further constrain the ith-dimensional head of each local structure to fall on the principle spine of its child structure.

$$H_{i+1} \subseteq \bigcup_{r \in R_i} [\mathrm{SP}_i(r, \mathcal{T})],$$

which is to say,

$$H_{i+1}(x) \rightarrow \mathrm{PSP}_i(x).$$

Fig. 7. Another 3-dimensional tree domain.

As there is only a single sequence of points in any 1-dimensional structure we need no further constraint on the interpretation of H_1. In fact, it is easy to see that the H_1 are just all points that are not roots in any dimension, i.e., $H_1 = T \setminus \bigcup_{1 \leq i \leq d}[R_i]$.

With such a distribution of H_i we can extend the $\bar{\lhd}_i$ throughout the structure in a way that will be preserved by the process of restriction to maximal points. This requires those relationships in which a point is dominated in a dimension i to be inherited by all points it dominates in dimensions greater than i and those relationships in which it dominates in dimension i to be inherited by those points it dominates in dimensions j greater than i *provided those points are on k-dimensional principle spines for some $i \leq k < j$.* In Figure 7, for example, (where heads have been marked by broadening the edge between them and their parent) we want the domination of $\langle\langle 1 \rangle\rangle$ by $\langle \varepsilon \rangle$ (in the second dimension) to be inherited by its descendants in the third dimension: $\langle\langle 1 \rangle, \varepsilon\rangle$, $\langle\langle 1 \rangle, \langle 0 \rangle\rangle$, $\langle\langle 1 \rangle, \langle 1 \rangle\rangle$, $\langle\langle 1 \rangle, \varepsilon, \varepsilon\rangle$, $\langle\langle 1 \rangle, \varepsilon, \langle 0 \rangle\rangle$ and $\langle\langle 1 \rangle, \varepsilon, \langle 1 \rangle\rangle$; and the the fact that $\langle\langle 1 \rangle\rangle$ dominates $\langle\langle 1 \rangle, \langle 1 \rangle\rangle$ (in the second dimension) to be inherited by $\langle\langle 1 \rangle, \varepsilon\rangle$, $\langle\langle 1 \rangle, \langle 0 \rangle\rangle$, $\langle\langle 1 \rangle, \varepsilon, \varepsilon\rangle$, and $\langle\langle 1 \rangle, \varepsilon, \langle 1 \rangle\rangle$ but not by $\langle\langle 1 \rangle, \langle 1 \rangle\rangle$ or $\langle\langle 1 \rangle, \varepsilon, \langle 0 \rangle\rangle$. Similarly, we want the domination of $\langle\langle 1 \rangle\rangle$ by $\langle\langle 0 \rangle\rangle$ (in the first dimension) *as well as* the fact that $\langle\langle 1 \rangle\rangle$ dominates $\langle\langle 2 \rangle\rangle$ (in the first dimension) to be inherited by *all* of its descendants in the second or third dimension: those above plus $\langle\langle 1, 0 \rangle\rangle$ and $\langle\langle 1, 1 \rangle\rangle$. None of these descendants are excluded since they are all on either 1-dimensional or 2 dimensional principle spines.

In the major dimension:

$$x \lhd_d^+ y \overset{\mathrm{def}}{\Longleftrightarrow} x \, \bar{\lhd}_d \, y$$

16

and, for each $1 \leq i \leq d$, $\vartriangleleft_i^+$ is the least relation on T satisfying:

$$x \, \bar{\vartriangleleft}_i \, y \Rightarrow x \, \vartriangleleft_i^+ \, y$$

$$(\exists y_1, \, i < j \leq d)[x \, \vartriangleleft_i^+ \, y_1 \text{ and } y_1 \, \vartriangleleft_j^* \, y] \Rightarrow x \, \vartriangleleft_i^+ \, y$$

$$(\exists x_1, \, i < j \leq d)[x_1 \, \vartriangleleft_j^* \, x \text{ and } x_1 \, \vartriangleleft_i^+ \, y \text{ and}$$

$$(\forall z)[x_1 \, \vartriangleleft_j^+ \, z \text{ and } z \, \vartriangleleft_j^* \, x \Rightarrow (\exists i \leq k < j)[\mathrm{PSP}_k(z)]]] \Rightarrow$$

$$x \, \vartriangleleft_i^+ \, y$$

where

$$x \, \vartriangleleft_j^* \, y \overset{\mathrm{def}}{\Longleftrightarrow} x \approx y \text{ or } x \, \vartriangleleft_j^+ \, y.$$

Again, these are monadic second-order definable relations (by defining a suitable class of paths).

Note that the interpretation of H_1 and the R_i is fixed by T, as is $\bar{\vartriangleleft}_i$ and $\vartriangleleft_d^+$, but, for all $i > 1$ the interpretation of H_i depends on the choice of H_j, $j < i$ and for all $i < d$ the interpretation of $\vartriangleleft_i^+$ depends on the choice of the H_j, $i \leq j < d$.

Again, the reason for using this structure is that we can now pick out the yield of a $\mathcal{T}$ as, roughly, a reduct of a substructure of it. To facilitate that, we will relax the definition to allow structures with arbitrary domains so long as they are isomorphic (wrt the $\vartriangleleft_i$) to a Σ-labeled headed Td.

6.2 The Yield Operation

We can now formally define the i-dimensional yield of a set of Σ-labeled headed Td. Let the dth-dimensional image of a point include that point plus the set of all points in the sequence of $(d-1)$-dimensional roots, ordered by $\vartriangleleft_d^+$, it dominates in the dth-dimension:

$$\mathrm{Image}_d(x) \overset{\mathrm{def}}{=} \{ y \mid x \, \vartriangleleft_d^* \, y \text{ and } (\forall z)[x \, \vartriangleleft_d^+ \, z \text{ and } z \, \vartriangleleft_d^* \, y \Rightarrow R_{d-1}(z)]]\}.$$

In Figure 7, for example,

$$\mathrm{Image}_d(\langle\langle 1 \rangle\rangle) = \{\langle\langle 1 \rangle\rangle, \langle\langle 1 \rangle, \varepsilon\rangle, \langle\langle 1 \rangle, \varepsilon, \varepsilon\rangle\}.$$

Let $\mathrm{Image}_d(X) = \bigcup_{x \in X}[\mathrm{Image}_d(x)]$.

17

The $(d-1)$-dimensional yield of a Td, $\mathcal{T}$, is the set of its maximal points wrt $\lhd_d^+$ ordered by $\lhd_{d-1}^+, \ldots, \lhd_1^+$. The $(d-1)^{\text{st}}$ -dimensional root of this structure is the maximal point, wrt $\lhd_d^+$, in the dth -dimensional image of the dth -dimensional root of $\mathcal{T}$—which is just the sole point in the intersection of that image and the domain of the yield—and the roots of the constituent structures are the maximal points in the dth -dimensional images of the roots of its corresponding constituent structures. We extend this to the heads as well: each H_i will be maximal points, wrt $\lhd_d^+$ of the dth -dimensional images of the H_i of $\mathcal{T}$.

Definition 7 *the ith -dimensional yield, for each $1 \leq i < d$, of a Σ-labeled headed Td* $\mathcal{T} = \langle T, \lhd_i^+, R_i, H_i, P_\sigma \rangle_{1 \leq i \leq d, \sigma \in \Sigma}$ *is*

$$Yield_d^i(\mathcal{T}) \stackrel{def}{=} \langle T^i, \lhd_j^+, R_j^i, H_j^i, P_\sigma^i \rangle_{1 \leq j \leq i, \sigma \in \Sigma}$$

defined as follows.

For $i = d - 1$:

$$T^{d-1} \stackrel{\text{def}}{=} \{x \in T \mid x \text{ is maximal wrt } \lhd_d^+\}.$$
$$R_{d-1}^{d-1} \stackrel{\text{def}}{=} \text{Image}_d(R_d) \cap T^{d-1}.$$
$$R_j^{d-1} \stackrel{\text{def}}{=} \text{Image}_d(R_j) \cap T^{d-1}, \; j < d - 1.$$
$$H_j^{d-1} \stackrel{\text{def}}{=} \text{Image}_d(H_j) \cap T^{d-1}, \; j \leq d - 1.$$
$$P_\sigma^{d-1} \stackrel{\text{def}}{=} P_\sigma \cap T^{d-1}.$$

For $1 \leq i < d - 1$:

$$\text{Yield}_d^i(\mathcal{T}) \stackrel{\text{def}}{=} \text{Yield}_{i+1}^i(\text{Yield}_d^{i+1}(\mathcal{T})).$$

Let

$$\text{Yield}_d^i(\mathbb{T}) = \{\text{Yield}_d^i(\mathcal{T}) \mid \mathcal{T} \in \mathbb{T}\}$$

Lemma 8 $Yield_j^i(Yield_k^j(\mathcal{T})) = Yield_k^i(\mathcal{T}))$.

This is immediate from the definition.

Note that, while the Σ-labeled headed Td are just a definable class of Σ-labeled Td, it is *not* the case that $\text{Yield}_d^i(\mathbb{T})$ is, in general, a definable set of Σ-labeled Ti. It is not difficult to show, for instance, that the string language

$$\{a_1^i a_2^i \cdots a_{2^{d-1}}^i \mid i > 1\}$$

18

is $\mathrm{Yield}_d^1(\mathbb{T})$ for some $\mathbb{T}$, a recognizable set of Td, but is not $\mathrm{Yield}_i^1(\mathbb{T})$ for any recognizable set of Ti for $i < d$ (and, in particular, for $d > 1$, is not a recognizable, hence definable, set of strings).

6.3   2-Branching Normal Form

A Td grammar or automaton is in *2-branching form* if the branching factor of each of its local Td is no greater than two. We can convert an arbitrary grammar or automaton into 2-branching form by iterating the familiar CNF-style transformation through each of its dimensions. As these transformations only add points that are non-maximal wrt the $\vartriangleleft_i^+$ and as they preserve the order of the original points wrt $\vartriangleleft_i^+$, they do not affect the one-dimensional yield.

Lemma 9 *A set $L \subset \Sigma^*$ is $\mathrm{Yield}_d^1(\mathbb{T})$ for $\mathbb{T}$, a set of Σ-labeled headed Td, iff it is $\mathrm{Yield}_d^1(\mathbb{T}')$ for $\mathbb{T}'$, a set of Σ-labeled headed Td that is in 2-branching form.*

6.4   Equivalence of Yields of Local and Recognizable Sets

Lemma 3 establishes the equivalence of local and recognizable sets modulo a projection. The proof hinges on the fact that the distinction between local and recognizable sets is solely due to the state information that is hidden in the recognizable sets; if this is made explicit as a component of the label, then recognizable sets become local. The projection simply strips this state information from the labels. As it turns out, in this construction there is no need to add state information to the labels of the maximal points. Hence, when dealing with the yields of these sets of structures we have a stronger result:

Lemma 10 *A set of Σ-labeled Td is the yield of a recognizable set of labeled Ti, for some $i > d$, iff it is the yield of a local set of labeled Ti.*

In the two-dimensional case, this just says that string languages yielded by the recognizable and local sets of trees coincide—they are just the CFLs [17,3].

7   wSnT3 and Tree-Adjoining Grammar

As noted above, the T2 grammars are equivalent to CFGs (with some mild generalizations) and the T2 automata are just finite-state tree automata. In
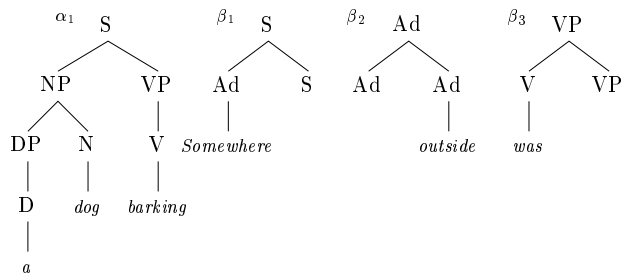
α_1 S β_1 S β_2 Ad β_3 VP

NP VP Ad S Ad Ad V VP

DP N V *Somewhere* *outside* *was*

D *dog* *barking*

a

Fig. 8. TAG elementary trees.

general, the process of concatenating local d-dimensional structures that is the underlying mechanism of the grammars and automata can be viewed, from the perspective of the $(d-1)$-dimensional yields, as a process of replacing a point—the root of the local Td—with a T$(d-1)$—the yield of the local Td. Thus, the concatenation of local trees corresponds to the substitution of strings for symbols—the string rewriting that is characteristic of Context-Free Grammars. At the 3-dimensional level, we have a corresponding process of substituting trees for nodes in trees, a particular form of context-free tree-rewriting. This process is the characteristic operation of *Tree-Adjoining Grammars*.

7.1 *Tree-Adjoining Grammars*

Definition 11 *A TAG [20–22] is a five-tuple* $\langle \Sigma, N, I, A, S \rangle$, *where:*

Σ *is the* terminal alphabet,
N *is the* non-terminal alphabet, $N \cap \Sigma = \emptyset$,
S *is the* start symbol, $S \in N$,
I *is a finite set of* initial trees *and*
A *is a finite set of* auxiliary trees.

Every non-frontier node of a tree in $I \cup A$ is labeled with a non-terminal. All frontier nodes are labeled with terminals with the exception that every tree in A has exactly one frontier node that is labeled with a non-terminal, its *foot*. This must be labeled with the same non-terminal as the root. The auxiliary and initial trees are distinguished by the presence (or absence, respectively) of a foot node. Together the initial and auxiliary trees are referred to as the *elementary* trees of the grammar. An example set of elementary trees is illustrated in Figure 8.

Derivation, in TAG, proceeds by a process of *adjunction* (see Figure 9) in which a node in one tree (the node at address η in the tree γ of the figure) is replaced by an auxiliary tree (β in the figure) by cutting out the subtree rooted at that node, attaching the auxiliary tree in its stead, and then attaching the excised subtree at the foot of the auxiliary tree. There is a clear parallel, here,
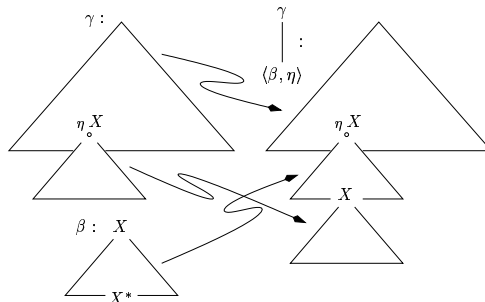
20

Fig. 9. Adjunction in TAG.

to context-free rewriting in strings. It differs from the general context-free tree rewriting of Rounds [23] in that all children of the rewritten node are attached as the children of a single node of the auxiliary tree with their order and number preserved.

In the original notion of TAG, all initial trees are rooted at a node labeled S. In *Lexicalized-TAG*, initial trees may be rooted in any non-terminal, non-terminals may occur anywhere in the frontier of the elementary trees, and there is a second combining operation, *substitution* in which an initial tree is attached at a non-terminal in the frontier of another tree. As substitution can be reduced to adjunction, we will restrict our attention to adjunction only, but we will admit initial trees rooted in non-terminals other than S. While derivations will involve a single initial tree, the derived structures may be partitioned into classes based on the label of the root of that tree.

In its pure form, the TAG derivation process is controlled exclusively by the labels of the nodes: an auxiliary tree may adjoin at a node iff its root bears the same label as that node. In practice, finer control is necessary and nodes may be associated with various *adjoining constraints*. A *null adjoining* constraint (NA) forbids adjunction at a node. An *obligatory adjoining* constraint (OA) requires it. A *selective adjoining* constraint (SA) is a set of names of auxiliary trees enumerating those which may be adjoined at that node. Note that NA constraints are subsumed by empty SA constraints.

Most linguistic uses of TAG employ a variation known as *Feature Structure Based TAG* (FTAG) [24] in which, rather than adjoining constraints, nodes are associated with feature structures (drawn from a finite set). These serve both to associate linguistic information with the node and to restrict the derivation process. The root and foot of auxiliary trees are each associated with single feature structures. Internal nodes are associated with two feature structures, a *top* and a *bottom* feature structure. In adjunction, the feature structure labeling the root of the auxiliary tree is required to unify with the top feature structure labeling the node of adjunction and the feature structure labeling its foot is required to unify with the bottom feature structure.

21

Selective adjoining constraints are realized by compatibility of the feature structures under unification. In the final derived structure the top and bottom feature structures of each node are required to unify. Thus, obligatory constraints can be enforced by labeling a node with incomparable top and bottom feature structures. Finally, null adjoining constraints can be realized by labeling a node with feature structures that fail to unify with any of those labeling the roots and feet of the auxiliary trees or by labeling the node with just a single feature structure, in essence unifying the top and bottom feature structure and blocking the adjunction mechanism.

7.2 Non-Strict TAGs

Having moved, now, to a mechanism in which the derivation process is controlled by the features of the nodes, it is no longer clear why the non-terminal labeling a node should have distinguished status. The requirement that the root and foot of an auxiliary tree be labeled with the same non-terminal and that it can only adjoin into similarly labeled nodes can be seen as a linguistic stipulation expressing the intention that auxiliary trees represent recursive fragments of phrase-structure trees—constituents that contain constituents of the same type. Similarly, one can view the feature-structure based restrictions as expressions of the linguistic analysis the grammar is intended to capture. For our purposes, it is useful to abstract away from these stipulative details. We will assume a fully general version of TAG in which the association between auxiliary trees and the nodes they may adjoin to is completely arbitrary.

Definition 12 *We will say that a TAG is* non-strict *if it permits the root and foot of auxiliary trees to differ in their label and to differ from the label of the nodes to which they may adjoin.*

Here we do not interpret the notion of label at all. All restrictions on adjunction must be expressed by explicit associations of some sort between the labels of nodes and the auxiliary trees which may adjoin at that node. Unless stated otherwise, we will assume these are stated as SA and OA constraints.

In non-strict TAGs (without substitution) there is no longer any meaningful distinction between terminals and non-terminals. Every internal node is labeled with a non-terminal, every leaf is a either a foot (in which case it is labeled with a non-terminal) or it is labeled with a terminal—the roles of the symbols can be unambiguously determined from the properties of the node they label. Thus, we can treat the fact that some labels represent elements of the target lexicon while others represent syntactic categories as an aspect of the theory of syntax a given grammar embodies and we will not distinguish terminals from non-terminals.

Under these circumstances, there is also no longer any reason to distinguish initial and auxiliary trees. While auxiliary trees are required to have foot nodes, since foot nodes are no longer required to be labeled with non-terminals there is no reason to prohibit initial trees from having foot nodes as well. The distinction between the roles of the trees is determined by the start symbol and the SA constraints. If a tree's root is labeled with the start symbol it may play the role of an initial tree. If it is named in an SA constraint, it may play the role of an auxiliary tree. Again, the fact that initial trees are intended to capture the minimal non-recursive structures of a language and the auxiliary trees are intended to capture minimal recursive structures can be seen as a stipulation that expresses an aspect of the intended theory of syntax the grammar embodies.

Finally, as the start symbol now serves only to pick out the set of permissible initial trees, we can drop it in favor of designating the (permissible) initial trees as a distinguished non-empty subset of the set of elementary trees. This is a generalization as it admits any non-empty finite set of start symbols, with the tree set and string language generated by a TAG with multiple start symbols being the union of those of the TAGs employing the start symbols individually. As both the TAG tree sets and the TAG string languages are closed under finite union, this does not change the generative capacity in any way.

Putting all this together, we will take a non-strict TAG to be simply a a pair $\langle E, I \rangle$ where E is a finite set of elementary trees in which each node is associated with:

- a label—drawn from some alphabet,
- an SA constraint—an arbitrary subset of the set of names of the elementary trees, and
- an OA constraint—Boolean valued

and $I \subseteq E$ is a distinguished non-empty subset, the initial trees. As every elementary tree named in an SA constraint is required to have a designated foot node, we will assume that each elementary tree has such a node.

7.3 Derivation Trees

A TAG *derivation tree* is a record of the adjunctions made in the course of a derivation. Its root is labeled with the name of an initial tree; all other nodes are labeled with a pair consisting of the name of an auxiliary tree and an address in the tree named in the parent of the node. In Figure 10, for instance, using the trees of Figure 8 and taking the derivation bottom-up: β_2 adjoins into β_1 at address $\langle 0 \rangle$ (or $\langle \varepsilon \rangle$), the resulting derived auxiliary tree
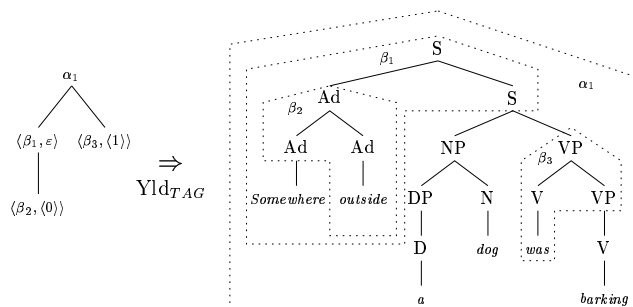
Fig. 10. Derivation and derived trees.

adjoins into α_1 at the root, and β_3 adjoins into α_1 at address $\langle 1 \rangle$.

As Weir [25] has pointed out, these derivation trees are context-free. In our case, in which the TAG is non-strict, a node $\langle \beta, w \rangle$ may be a child of another node $\langle \gamma, v \rangle$ iff the SA constraint associated with w in γ admits β; and if the address w in γ is associated with an OA constraint then any node labeled $\langle \gamma, v \rangle$ is required to have a child labeled $\langle \beta, w \rangle$ for some β.

The derived tree is obtained from the derivation tree by a *yield* operation which simultaneously applies the specified adjunctions. The tree set generated by a TAG G, denoted $T(G)$, is the (tree) yield of the derivation trees it licenses. The string language of G, denoted $L(G)$, is the string yield of that tree set.

7.4 Equivalence of T3-Automata and Non-Strict TAGs

The equivalence of T3-automata and non-strict TAGs can be established by straightforward constructions—in essence, SA constraints and states have roughly equivalent power. In the following section we will sketch the interpretation of an FTAG grammar in wSnT3. As definability is equivalent to recognizability and the feature system of FTAG is at least as powerful as any of the other systems of adjoining constraints, this serves to establish that every TAG tree set is the two-dimensional yield of a recognizable set of Σ-labeled T3.

In this section we will sketch the construction of a non-strict TAG with explicit SA and OA constraints which derives the two-dimensional yield of a local set of Σ-labeled T3. As a set of Σ-labeled Td is the yield of a local set iff it is the yield of a recognizable set (Lemma 10) this will establish that every set of trees that is the yield of a recognizable set of Σ-labeled T3 is a TAG tree set for a non-strict TAG with, at least, SA and OA constraints. Together, these give us our central theorem:

Theorem 13 *A set of Σ-labeled trees is the yield of a recognizable set of Σ-labeled T3 iff it is generated by a non-strict TAG with adjoining constraints.*

24

Proof (of the forward direction) To show that the tree yield of a recognizable set of Σ-labeled T3 can be generated by a non-strict TAG, as we noted in the introductory remarks of this section, it suffices to show how to construct a TAG generating the tree yield of a local set of Σ-labeled T3. Suppose, then, that

$$\mathbb{T} = \mathrm{Yield}_3^2(\mathcal{G}(\Sigma_0))$$

for some T3 grammar $\mathcal{G}$ and set of initial labels Σ_0. We will construct a non-strict TAG $G_\mathcal{G} = \langle E_\mathcal{G}, I_\mathcal{G}\rangle$ with SA and OA constraints such that $\mathbb{T}$ is $T(G_\mathcal{G})$.

Let $G_\mathcal{G}$ be a non-strict TAG with elementary tree set:

$$E_\mathcal{G} \stackrel{\mathrm{def}}{=} \{\mathcal{T} \mid \langle \sigma, \mathcal{T}\rangle \in \mathcal{G}, \text{ for some } \sigma\}.$$

The foot of each $\mathcal{T} \in E_\mathcal{G}$ is just the maximal point of its principle spine.

For each $\mathcal{T} = \langle T, \tau\rangle \in E_\mathcal{G}$ and each $w \in T$, let

$$\mathrm{SA}_{\langle \mathcal{T}, w\rangle} \stackrel{\mathrm{def}}{=} \{\mathcal{T}' \mid \langle \tau(w), \mathcal{T}'\rangle \in \mathcal{G}\}.$$

and let

$$\mathrm{OA}_{\langle \mathcal{T}, w\rangle} \stackrel{\mathrm{def}}{=} \langle \tau(w), \emptyset\rangle \notin \mathcal{G}.$$

Which is to say that the SA constraint of w in $\mathcal{T}$ includes every tree that is the yield of a local T3 in $\mathcal{G}$ with root labeled the same as w and that the w in $\mathcal{T}$ which bear obligatory adjoining constraints are just those which are not licensed to be maximal.

Finally, let the set of initial trees of $G_\mathcal{G}$ be

$$I_\mathcal{G} \stackrel{\mathrm{def}}{=} \{\mathcal{T} \mid \langle \sigma, \mathcal{T}\rangle \in \mathcal{G}, \text{ for some } \sigma \in \Sigma_0\}.$$

The equivalence of $G_\mathcal{G}$ and $\mathcal{G}$ hinges on the fact that there is a direct correspondence between the Σ-labeled T3 in $\mathcal{G}(\Sigma_0)$ and the derivation trees of $G_\mathcal{G}$ which is witnessed by a map that takes the local T3 of the one to the nodes of the other. For any $\mathcal{T} = \langle T, \tau\rangle \in \mathcal{G}(\Sigma_0)$ and $s \in T$ let $\mathcal{T}_s$ denote the local tree rooted at address s of $\mathcal{T}$. This will have been licensed by some production $\langle \tau(s), \langle T, \tau\rangle|_{\mathrm{Ch}(T, s)}\rangle \in \mathcal{G}$. Let f map $\mathcal{T}_\varepsilon$ to a node labeled (with the name of) $\langle T, \tau\rangle|_{\mathrm{Ch}(T, \varepsilon)}$ and, for each $s \cdot \langle w\rangle \in T$, let f map $\mathcal{T}_{s \cdot \langle w\rangle}$ to a node labeled $\langle \langle T, \tau\rangle|_{\mathrm{Ch}(T, s \cdot \langle w\rangle)}, w\rangle$. Finally let $f(\mathcal{T}_s)$ be the parent of $f(\mathcal{T}_{s \cdot \langle w\rangle})$.

25

Since $\mathcal{T}$ is a T3 (and, therefore, "treelike" in its third dimension) the image of $\mathcal{T}$ under f forms a tree. Since the trees associated with the nodes in the image of $\mathcal{T}$ under f are all component trees of $\mathcal{T}$, they are all included in $E_{\mathcal{G}}$, the elementary trees of $G_{\mathcal{G}}$. Since $\mathcal{T} \in \mathcal{G}(\Sigma_0)$, each $\mathcal{T}_s$ is a local tree in $\mathcal{G}$. Consequently, for all $s \cdot \langle w \rangle$, the (name of the) tree $\langle T, \tau \rangle |_{\mathrm{Ch}(T, s \cdot \langle w \rangle)}$ will be an element of $\mathrm{SA}_{\langle \langle T, \tau \rangle |_{\mathrm{Ch}(T,s)}, w \rangle}$. In other words, if a node is labeled $\langle \mathcal{T}', w \rangle$ in the image of $\mathcal{T}$ then the tree $\mathcal{T}'$ is licensed to adjoin into the tree associated with its parent by the SA constraint associated with the node at address w in that parent tree. Moreover, if $\mathcal{T}_{s \cdot \langle w \rangle}$ is trivial (i.e., $s \cdot \langle w \rangle$ is maximal wrt $\lhd_3$) then the OA constraint associated with w in the component tree containing it is false. Finally, since the root of $\mathcal{T}_{\varepsilon}$ must be labeled with a member of Σ_0 and the root of the image of $\mathcal{T}$ under f is the image of $\mathcal{T}_{\varepsilon}$, the root of the image of $\mathcal{T}$ under f is labeled with (the name of) an initial tree of $G_{\mathcal{G}}$.

In other words, the image of $\mathcal{T}$ under f is a derivation tree of $G_{\mathcal{G}}$. A similar analyses establishes that if a derivation tree of $G_{\mathcal{G}}$ is the image of a Σ-labeled T3 $\mathcal{T}$ then $\mathcal{T} \in \mathcal{G}(\Sigma_0)$.

All that remains is to establish that the TAG yield of the $f(\mathcal{T})$ is $\mathrm{Yield}_3^2(\mathcal{T})$. This is simply a matter of checking that their domains are equal and that $\lhd_2^+$ and $\lhd_1^+$ in $\mathcal{T}$ are equal to proper domination and linear precedence in the derived tree. The equality of domains follows from the fact that a point $s \cdot \langle w \rangle$ is maximal (wrt $\lhd_3$) in $\mathcal{T}$ iff $f(\mathcal{T}_s)$ has no child labeled $\langle \mathcal{T}', w \rangle$ iff no tree adjoins at w in the tree associated with $f(\mathcal{T})$ when taking the TAG yield.

The equality of the relations can be establish by analyzing the cases of the definition of $\lhd_i^+$. We will look only at $\lhd_2^+$ and proper domination. The analysis for $\lhd_1^+$ and linear precedence is similar. If $x \lhd_2^+ y$ then either:

- $x \, \bar{\lhd}_2 \, y$, in which case x and y occur in the same elementary tree of $G_{\mathcal{G}}$ with x dominating y, or
- $x \lhd_2^+ y_1$ and $y_1 \lhd_3^* y$ for some y_1 in which case x dominates y_1 (by induction) and y occurs in an elementary tree which will eventually adjoin (possibly in a derived auxiliary tree) at y_1, or
- $x_1 \lhd_3^* x$ and $x_1 \lhd_2^+ y$ for some x_1 and all z falling between x_1 (exclusive) and x (inclusive) wrt $\lhd_3^*$ are on the principle spine of their component tree, in which case x occurs in an elementary tree that will eventually adjoin at x_1 with x dominating the root of that elementary tree (and that of every intermediate derived elementary tree), and that root will dominate y (again by induction).

$\square$

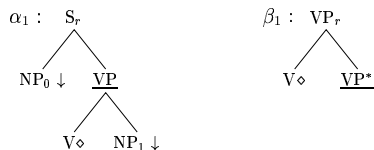It is important to recognize just how direct this construction is. In a very
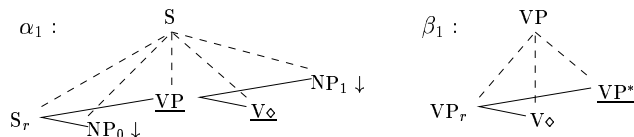
Fig. 11. XTAG verb trees.

Fig. 12. XTAG initial trees as local T3

strong sense the T3 grammar and the TAG are just alternate presentations of the same object, with the local relationships of the grammar being expressed in the SA constraints of the TAG. Moreover, the choice of grammars rather than automata as a starting point is purely a matter of technical convenience. The relationships expressed by the SA constraints are independent of the labels of the nodes in precisely the same way that the Q-labeled local trees of the automata are. In essence, states and SA constraints are equivalent mechanisms; for all intents and purposes T3 automata and non-strict TAGs with adjoining constraints are just notational variants.

8   TAG as Sets of Logical Constraints

The practical value of the fact that the tree and string languages definable in wSnT3 coincide with the TAG tree sets and TALs is that we can define our syntactic analysis in abstract logical terms and then compile them into T3 automata, equivalently TAGs. As an example of how this works out, we will look at capturing a fragment of an existing FTAG grammar, the XTAG grammar of [26]. We will look, in particular at the handling of case assignment in XTAG main verb (α_1) and auxiliary verb (β_1) trees (Figure 11). [4]

The basic idea is to interpret node names as first-order variables and tree names as monadic second-order variables with, e.g., $\alpha_1(x)$ satisfied iff x is the

---

[4] Here the nodes marked with $\downarrow$ are substitution nodes; we will treat substitution simply as adjunction at a leaf. Those marked with $\diamond$ are *anchor* nodes—linguistically, each elementary tree is an extended projection of its anchor.
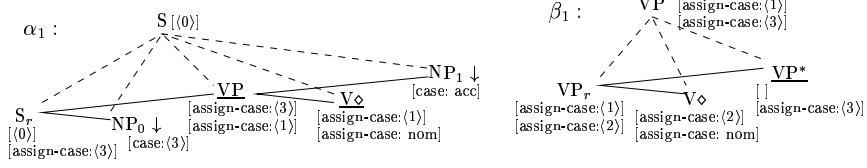
Fig. 13. Case assignment in XTAG.

(3rd -dimensional) root of the local T3 corresponding to α_1:

$\alpha_1(x) \leftrightarrow$

$(\exists s_r, np_0, vp, v, np_1)[x \lhd_3 s_r \wedge x \lhd_3 np_0 \wedge x \lhd_3 vp \wedge x \lhd_3 v \wedge x \lhd_3 np_1 \wedge$

$\text{Min}_2(s_r) \wedge \text{Max}_2(np_0) \wedge \text{Max}_2(v) \wedge \text{Max}_2(np_1) \wedge$

$s_r \lhd_2 np_0 \wedge s_r \lhd_2 vp \wedge \text{H}_1(vp) \wedge \text{Min}_1(np_0) \wedge np_0 \lhd_1 vp \wedge \text{Max}_1(vp) \wedge$

$vp \lhd_2 v \wedge vp \lhd_2 np_1 \wedge \text{H}_1(v) \wedge \text{Min}_1(v) \wedge v \lhd_1 np_1 \wedge \text{Max}_1(np_1) \wedge$

$\text{Initial}(x) \wedge \text{Anchor}(v) \wedge \text{Subst}(np_0) \wedge \text{Subst}(np_1) \qquad]$

Here Min_i and Max_i pick out minimal (root) and maximal (leaf) nodes wrt the ith dimension—these are defined predicates. $\text{Initial}(x)$ is true at the root of each local T3 encoding an initial tree, $\text{Anchor}(x)$ is true at each anchor node (we will ignore insertion of the lexical items), and $\text{Subst}(x)$ is true at each node marked for substitution—these are labels, in Σ. It is possible to treat substitution as concatenation of trees but simpler to treat it as adjunction. We require all Subst nodes to have children in the 3rd -dimension and require the set of Initial nodes to be exactly the Subst nodes plus the root of the entire T3:

$$(\forall x)[\text{Subst}(x) \rightarrow (\exists y)[x \lhd_3 y]] \quad (\forall x)[\text{Initial}(x) \leftrightarrow (\text{Subst}(x) \vee \text{Min}_3(x))]$$

Figure 13 shows the distribution of features responsible for case assignment in the XTAG grammar. Following the approach of [10] we interpret the paths occurring in the feature structures decorating the trees as monadic predicates: Σ includes each sequence of features that is a prefix of a path occurring in a feature-structure derivable in the grammar. We will refer to this set of sequences as Feat. Each node is multiply labeled: the feature-structure associated with it is the union of the paths labeling it. In order to capture the distinction between top and bottom feature-structures we will prefix their paths with 't' and 'b', respectively. We can then add to the definition of α_1:

$$\langle t : case : acc \rangle (np_1) \wedge \langle b : assign\text{-}case : nom \rangle (v).$$

This encoding of feature-structures as sets of paths gives us a straightforward

28

definition of predicates for path equations as well. For any sequences $w, v \in$ Feat:

$$\langle w = v \rangle (x, y) \equiv \bigwedge_{\substack{w:u \in \mathrm{Feat} \\ \text{or } v:u \in \mathrm{Feat}}} [\langle w : u \rangle (x) \leftrightarrow \langle v : u \rangle (y)].$$

With this we can add the re-entrancy tags:

$$\langle \mathrm{b} : \text{assign-case} = \mathrm{t} : \text{assign-case} \rangle (vp, v) \wedge$$

$$\langle \mathrm{b} : \text{assign-case} = \mathrm{t} : \text{case} \rangle (s_r, np_0) \wedge$$

$$\langle \mathrm{b} : \text{assign-case} = \mathrm{t} : \text{assign-case} \rangle (s_r, vp) \wedge \langle \mathrm{b} = \mathrm{t} \rangle (s, s_r).$$

The labeling of the elementary trees can then be interpreted as a collection of constraints on local T3, with the set of structures licensed by the grammar being the set of T3 in which every node satisfies one of these collections of constraints. Note that for a T3 in which the β_1 T3 expands the VP node in an α_1 T3 to be licensed, the VP node must satisfy both the constraints of the α_1 T3 and the constraints on the root of the β_1 T3. Thus the top feature-structure of the VP is unified with the top feature-structure of VP_r and the bottom feature-structure with the bottom feature-structure of the foot VP by simple transitivity of equality. There is no need for additional path equations and no extra-logical mechanisms of any sort; licensing is simply a matter of ordinary model-theoretic satisfaction. To get the (default) unification of top and bottom feature structures of nodes that are not expanded by adjunction we add a single universal principle:

$$(\forall x)[\mathrm{Max}_3(x) \rightarrow \langle t = b \rangle (x, x)].$$

Taken literally, this approach yields little more than a fully declarative restatement of the original grammar. But, in fact, a large proportion of the features decorating elementary trees are there only to facilitate the transport of features through the tree: there is no obvious linguistic motivation for positing that "assign-case" is a feature of VPs or of S. In the language of wSnT3 there is no need for these intermediate features or even any need to distinguish top and bottom feature structures—we can state directly that the value of the *case* feature of the subject NP, for instance, must agree with the value of the *assign-case* feature of the verb. Of course, what is interesting about this relationship is the effect of adjoined auxiliaries. The TAG analysis includes an *assign-case* feature for the intermediate VP in order to allow auxiliary verbs adjoined at the VP to intercept this relationship by interposing between the VP's top and bottom feature structures. In wSnT3 we obtain the same result from the way in which we identify the relevant verb. For instance, if we take

29

it to be the last adjoined verb[5] —the one most deeply embedded in the third dimension—we can add to the definition of α_1:

$$(\exists y, z)[vp \lhd_3^* y \land \mathrm{Max}_3(y) \land y \, \bar{\lhd}_2 \, z \land \langle\text{assign-case}\rangle(z) \land$$
$$\langle\text{assign-case} = \text{case}\rangle(z, np_0)].$$

Having liberated the definition of the well-formed syntactic structures from the needs of the grammar mechanism there is no reason to limit ourselves to the structural relationships that the mechanism employs. Rather, we are free to state the theory directly in terms of any linguistically significant relationship (Government, for example) that is definable within wSnT3—the grammar, in effect, becomes a direct expression of the linguistic theories it is intended to incorporate.

It is worth noting that, following typical desiderata for contemporary theories of syntax, one expects the theory to be expressed as the consequences of a moderate number of relatively simple principles, interpreted conjunctively. From the perspective of the translation into an automaton, i.e., a TAG, this corresponds to generating the set of elementary trees by the interaction of a set of partially defined structures—in effect, to a factorization of the grammar into a relatively small set of fragmentary components. Systems of this sort are an active area of research in the TAG community, where they are seen as a way of taming the complexity of defining and maintaining the large sets of trees needed for wide-coverage TAGs [27–30]. From our current perspective, the practical motivations of the grammar writers are not so very different than the (meta-)theoretic motivations of the syntacticians.

Putting it all together, then, one gets the prospect of a large-scale grammar in which all aspects of the set of elementary trees are the consequences of an interacting set of relatively simple and highly modular principles, one that can be developed and maintained directly in terms of these abstract principles with the actual expansion into elementary trees being carried out by the automaton construction algorithm.

9   Practical Issues

As we suggested earlier, the weakness in this program is the extraordinary conciseness of the logical formulae relative to the automata/TAGs. Viewed from another perspective, the practical application of the automaton construction

---

[5] This is correct only if the foot nodes have null-adjoining constraints, as is usual.

algorithm is limited by its non-elementary complexity. As it turns out, however, experience in the one-dimensional case [7,31] has shown that, in many useful cases, the construction is reasonably well-behaved. The problem, then, is to find ways of avoiding the potential inefficiency—to find fragments of the full MSO languages that do not suffer from the hyper-exponential blow-up.

Morawietz and Cornell [12,13] have explored these issues in the context of applying the two-dimensional construction to encodings of GB-style principles. There are certain obvious things one can do in order to help limit the complexity: One should limit the total number of free variables employed. (The size of the label set is exponential in the number of free variables). One should limit the quantifier depth. (The asymptotic size of the state set is a function of the number of quantifier alternations.) Most importantly, as a practical matter, one needs to limit the overall size of the formulae one works with. This last fact has led them to embed the automata construction as the constraint solver in a constraint logic-programming context—individual constraints are relatively tractable and the CLP framework can be used to combine their effects.

There are a number of characteristics of the TAG application that suggest that, despite its higher dimension, it may yet be at least as tractable as the GB application. Chief among these is the fact that, because, in the TAG analysis, elementary trees are required to include the entire subcategorization frame, many relationships that are handled through the mediation of indices in GB are local to the elementary structures. It is the indexation mechanism that seems to push GB-style accounts over the line between the context-free and the context-sensitive [9] and this mechanism, in its bounded form, turned out to be a particularly intractable component in Morawietz and Cornell's work. In effect, the indexation mechanism needs to allow all ways (or in the bounded form, all ways up to some fixed bound on the number of classes) of partitioning the nodes of the structure into equivalence classes. It is hardly surprising that this should be difficult. The elimination of arbitrary indexation in the TAG account, in addition to being theoretically attractive, may help to eliminate some of the limiting aspects of the GB theories.

The factorizations such as those mentioned in the previous section also should help to control the complexity. The finer the factorization of the theory, the smaller the formulae required to define each component. As the constructions for conjunction and disjunction of automata are quite efficient, it should be possible to compile the components separately, much as Morawietz and Cornell's CLP system solves them as separate constraints, and then combine the results without suffering excessively large automata at intermediate stages.

Finally, many of the structural principles of existing factorizations of TAGs are effectively constraints on the structure of the elementary trees. Given that, it may be possible to implement these as filters on the sets of structures produced

31

by the other principles of the grammar in a way that is even more efficient than the simple cross-product construction for conjunction.

10   Conclusion

We have presented a generalization of well known results that characterize language-theoretic complexity classes in terms of definability in the weak monadic second-order theories of strings and trees to tree-like structures of arbitrary dimension. At the third level this gives us a characterization of the Tree-Adjoining Languages in terms of definability in the weak monadic second-order theory of three dimensional tree domains. We have looked briefly at the potential for using this result to provide a sort of logic-programming for TAGs which would subsume many of the current schemes to simplify large scale TAG development and maintenance. While this program suffers from the same potential intractability problems as all applications of the MSO constructions, there is reason to believe that it may well be amenable to careful engineering. One way or the other, these issues will be resolved as the program is completed.

References

[1] J. R. Büchi, Weak second-order arithmetic and finite automata, Zeitschrift für mathematische Logik und Grundlagen der Mathematik 6 (1960) 66–92.

[2] C. C. Elgot, Decision problems of finite automata design and related arithmetics, Transactions of the American Mathematical Society 98 (1961) 21–51.

[3] J. Doner, Tree acceptors and some of their applications, Journal of Computer and System Sciences 4 (1970) 406–451.

[4] J. W. Thatcher, J. B. Wright, Generalized finite automata theory with an application to a decision problem of second-order logic, Mathematical Systems Theory 2 (1) (1968) 57–81.

[5] M. O. Rabin, Decidability of second-order theories and automata on infinite trees, Transactions of the American Mathematical Society 141 (1969) 1–35.

[6] A. Ayari, D. Basin, S. Friedrich, Structural and behavioral modeling with monadic logics, in: R. Drechsler, B. Becker (Eds.), The Twenty-Ninth IEEE International Symposium on Multiple-Valued Logic, IEEE Computer Society, Los Alamitos, Freiburg, Germany, 1999, pp. 142–151.

[7] D. Basin, N. Klarlund, Hardware verification using monadic second-order logic, in: Computer Aided Verification: 7th International Conference, CAV'95, no. 939 in Lecture Notes in Computer Science, Springer, 1995.

[8] P. Kelb, T. Margaria, M. Mendler, C. Gsottberger, MOSEL: A flexible toolset for monadic second-order logic, in: E. Brinksma (Ed.), Tools and Algorithms for the Construction and Analysis of Systems, TACAS '97, LNCS 1217, Enschede, The Netherlands, 1997.

[9] J. Rogers, A Descriptive Approach to Language-Theoretic Complexity, Studies in Logic, Language, and Information, CSLI/FoLLI, 1998.

[10] J. Rogers, "Grammarless" phrase structure grammar, Linguistics and Philosophy 20 (1997) 721–746.

[11] J. Rogers, A model-theoretic framework for theories of syntax, in: Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, Santa Cruz, CA, 1996, pp. 10–16.

[12] F. Morawietz, T. Cornell, Representing constraints with automata, in: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, Madrid, Spain, 1997, pp. 468–475.

[13] F. Morawietz, T. Cornell, The MSO logic-automaton connection in linguistics, in: Logical Aspects of Computational Lingistics, 1998.

[14] D. J. Weir, A geometric hierarchy beyond context-free languages, Theoretical Computer Science 104 (1992) 235–261.

[15] S. Gorn, Processors for infinite codes of Shannon-Fano type, in: Symp. Math. Theory of Automata, 1962.

[16] N. Chomsky, M. P. Schüzenberger, The algebraic theory of context-free languages, in: P. Braffort, D. Hirschberg (Eds.), Computer Programming and Formal Systems, 2nd Edition, Studies in Logic and the Foundations of Mathematics, North-Holland, Amsterdam, 1963, pp. 118–161.

[17] J. W. Thatcher, Characterizing derivation trees of context-free grammars through a generalization of finite automata theory, Journal of Computer and System Sciences 1 (1967) 317–322.

[18] F. Gécseg, M. Steinby, Tree languages, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages: Beyond Words, Vol. 3, Springer, 1997, pp. 1–68.

[19] A. R. Meyer, Weak monadic second order theory of successor is not elementary-recursive., in: Proceedings, Logic Colloquium, no. 3 in Lecture Notes in Mathematics, Springer, 1975, pp. 132–154.

[20] A. K. Joshi, L. S. Levy, M. Takahashi, Tree adjunct grammars, Journal of Computer and System Sciences 10 (1975) 136–163.

[21] A. K. Joshi, How much context-sensitivity is required to provide reasonable structural descriptions: Tree adjoining grammars, in: D. Dowty, L. Karttunen, A. Zwicky (Eds.), Natural Language Processing: Psycholinguistic, Computational and Theoretical Perspectives, Cambridge University Press, 1985.

[22] A. K. Joshi, Y. Schabes, Tree-adjoining grammars and lexicalized grammars, in: M. Nivat, A. Podelski (Eds.), Tree Automata and Languages, Elsevier Science Publishers B.V., 1992, pp. 409–431.

[23] W. C. Rounds, Mappings and grammars on trees, Mathematical Systems Theory 4 (1970) 257–287.

[24] K. Vijay-Shanker, A. K. Joshi, Unification based tree adjoining grammars, in: J. Wedekind (Ed.), Unification-based Grammars, MIT Press, Cambridge, MA, 1991.

[25] D. J. Weir, Characterizing mildly context-sensitive grammar formalisms, Ph.D. thesis, University of Pennsylvania (1988).

[26] XTAG Research Group, A lexicalized tree adjoining grammar for English, Tech. Rep. IRCS-98-18, Institute for Research in Cognitive Science (1998).

[27] T. Becker, HyTAG: A new type of tree adjoining grammars for hybrid syntactic representation of free word order langauges, Ph.D. thesis, Universität des Saarlandes, Saarbrücken (1993).

[28] K. Vijay-Shanker, Y. Schabes, Structure sharing in lexicalized tree-adjoining grammars, in: Proceedings COLING'92, 1992.

[29] M.-H. Candito, A principle-based hierarchical representation of LTAGs, in: 16th International Conference on Computational Linguistics, COLING'96, 1996.

[30] R. Evans, G. Gazdar, D. Weir, Encoding lexicalized tree adjoining grammars with a nonmonotonic inheritance hierarchy, in: 33rd Annual Meeting of the Association for Computational Linguistics, Cambridge, MA, 1995, pp. 77–84.

[31] J. G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, R. Paige, T. Rauhe, A. Sandhol, MONA: Monadic second-order logic in practice, in: E. Brinksma (Ed.), Tools and Algorithms for the Construction and Analysis of Systems, TACAS'95 Selected Papers, no. 1019 in Lecture Notes in Computer Science, Springer, 1995, pp. 89–110.