Syntactic Structures as Multi-dimensional Trees

JAMES ROGERS

Department of Computer Science, Earlham College, Richmond, IN, 47374, USA, E-mail: jrogers@cs.ucf.edu

ABSTRACT: We survey a sequence of results relating model-theoretic and language-theoretic definability over an infinite hierarchy of multi-dimensional tree-like structures and explore their applications to a corresponding range of theories of syntax. We discuss, in particular, results for Government and Binding Theory (GB), Tree-Adjoining Grammar (TAG) and Generalized Phrase-Structure Grammar (GPSG) along with a generalized version of TAG extending TAG in much the same way that GPSG extends CFLs. In addition, we look at a hierarchy of language classes, Weir's version of the Control Language Hierarchy, which is characterized by definability in our hierarchy and speculate on possible linguistic significance of higher levels of these hierarchies.

KEYWORDS: Syntax, trees, wMSO theories, grammars, automata, GB, TAG, GPSG, control languages

1 Introduction

The goal of this paper is to survey a sequence of formal results along with a parallel sequence of their application to issues in Computational Linguistics. The origins of the formal results date back to work by Büchi [3] and Elgot [10] in the early '60's which *inter alia* established a descriptive characterization of the regular (string) languages—a characterization of the regular languages in terms of logical definability over a particular class of model-theoretic structures: a set of (finite) strings is regular iff it is definable, in a particular sense, within the weak monadic second-order theory of the natural numbers with successor (wS1S). This was extended ten years later by Doner [9] and Thatcher and Wright [44] to a similar descriptive characterization of the recognizable sets of trees and, consequently, the context-free (string) languages: a set of strings is context-free iff it is the yield of a set of finite labeled trees that is definable in the weak monadic second-order theory of multiple successors (wSnS). More recently, we have generalized these results to obtain a characterization of the Tree-Adjoining Languages [19, 21] in terms of a class of three-dimensional trees, and thence to an infinite hierarchy of structures which serve to characterize a parallel hierarchy of languages: Weir's Control Language Hierarchy [47].

From a technical perspective the formal results are interesting, in part, because the involve application of automata theory to questions in pure model theory. The results for regular and context-free languages were obtained on the way to establishing decidability of the full monadic second-order theories. (The result for the full MSO theory of multiple successors, SnS, was obtained by Rabin [27].) The technique of the proof is to construct an automaton that accepts all and only the set of satisfying assignments (represented as labeled structures) for a given formula in the appropriate logical language. A formula, then, is satisfiable iff the set recognized by the corresponding automaton is non-empty. It remains only to establish the decidability of emptiness for these recognizable sets.

In this way the formal results, in a sense, borrow results from Formal Language Theory to obtain results in model theory. In doing so, they establish a connection between logical definability and formal languages. From the perspective Formal Language Theory and, in particular, Computational Linguistics what is attractive about this is the connection it can provide between declarative (constraint- or principle-based) mechanisms for defining languages and generative (grammar- and automata-based) mechanisms. This connection not only provides a way of establishing language-theoretic complexity results for declarative accounts of syntax but, more importantly, provides means of comparing accounts in a theory-neutral context, of importing fragments of one account into another and, at least potentially, of developing computational mechanisms for processing the declarative accounts. In this way, by exploiting the connection between definability and recognizability in the opposite direc-

tion we, in a sense, collect on the original debt.

Our focus, here, is not the formal results themselves nor even the details of their applications. Rather, our intention is to give a sense of the range of the results and, in particular, the way in which incremental extensions in the model-theoretic domain correspond to increments in the power of the generative mechanisms in the language-theoretic domain. Consequently, our treatment is at quite a high level: we omit nearly all the proofs and many of the formal details and provide, in their stead, references to papers in which the details can be found.

In the next section we introduce our hierarchy of multi-dimensional trees, along with their wMSO theories, grammars and automata, and sketch the equivalences between these. The rest of the paper is organized as a tour in which we visit classes of structures with increasing complexity, sketching their application to a corresponding range of linguistic theories as we go. We start, in Sections 3 and 5 with structures with finitely bounded branching, first in two dimensions where we explore applications to Government and Binding Theory, and then, after pausing, in 4, to sketch a notion of string yield for higher-dimensional trees, we move to three dimensions, where we explore applications to Tree-Adjoining Grammar. In Section 6 we extend our results to the ω-branching structures. This allows us to treat, in two dimensions, GPSG (Section 7) and, in three dimensions, a similarly generalized version of TAG (Section 8). In Section 9 we bring it back together by sketching the equivalence of the hierarchy as a whole to Weir's hierarchy of control grammars. We then close with some thoughts about the nature of the relationships between the model-theoretic and language-theoretic results along with a glimpse of the ongoing work in this area.

2 Multi-Dimensional Trees

The structures Büchi and Elgot used to model strings are just finite initial segments of the natural numbers, ordered by successor and labeled with a finite set of monadic predicates (the symbols of the alphabet). Doner, Thatcher and Wright, and Rabin lifted these structures to ordered, rooted trees by adopting multiple successor functions indexed by natural numbers. The immediate successors of a point wrt these functions are its children in the tree, with the left-to-right ordering of the children being imposed by the indices of the successor functions. In a strong sense, this is as far as this approach goes. As we will see shortly, for all n greater than one, the n-successor structures are equivalent in the sense that they can all be embedded in the two-successor structure. (This includes the ω-successor structure.)

In order to generate a useful hierarchy with additional successors we must interpret the successors differently. We take the children of a node to be a
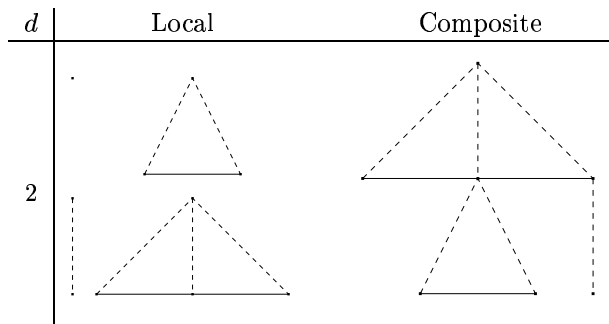
FIG. 1. Some 2-dimensional trees.

FIG. 2. Some 0- and 1-dimensional trees.

structure of the sort employed by Büchi and Elgot. We build *local trees* (trees of depth at most one) by adding a single point and connecting it to each of the nodes in the child structure by a single second-dimensional successor relation. Thus the left-to-right ordering of the children is now explicit in the first-dimensional successor rather than being implied by the ordering of multiple second-dimensional successors.. We refer to the adjoined point as the *root* of the local tree and the child structure as its *yield*. We include, as local trees, both the empty tree (with neither root nor yield) and the trivial tree in which the yield is empty. Trees of depth greater than one (*composite trees*) are constructed by joining local trees, identifying the root of one with some point in the yield of another. (See Figure 1.)

We can generalize this both downward, to the string—and even point—structures, and upward to structures of arbitrary dimension. At the one-dimensional (string) level the local structures are just pairs of points (or possibly a point by itself—a trivial string) related by the first-dimensional successor, one being the root the other the yield. These combine in the same way as the local trees, identifying the root of one local string with the yield of another,
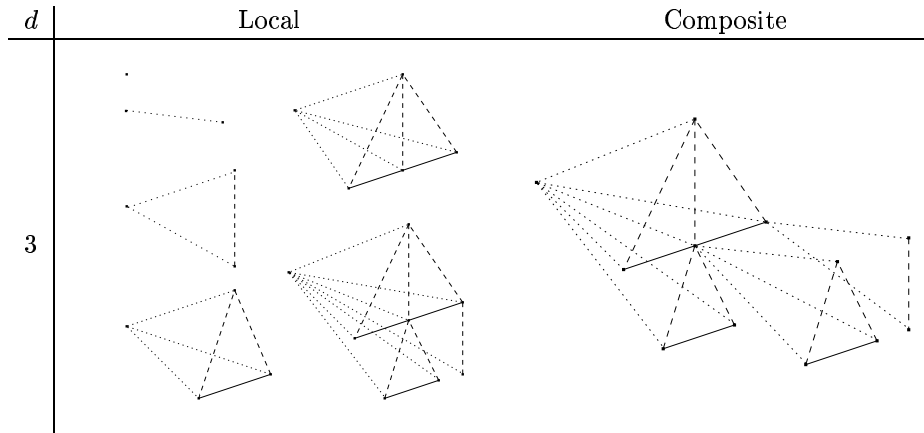
d	Local	Composite

FIG. 3. Some 3-dimensional trees.

to provide structures of arbitrary depth (Figure 2.) At this level, the process of building composite structures out of local structures is ordinary *concatenation* of strings. We adopt this terminology uniformly and refer to the process of identifying the root of a local tree with a point in the yield of another as concatenation of trees.

At the zero-dimensional level, we have just roots with no successors. Note that just as the local trees are built by adding a second-dimensional root to an arbitrary one-dimensional structure, the local strings are built by adding a first-dimensional root to an (arbitrary) zero-dimensional structure. To generalize in the upward direction we simply iterate this. We build local three-dimensional trees by adding a single point (a third-dimensional root), related by single third-dimensional successor relation to an arbitrary tree structure. Composite three-dimensional trees are built by concatenating these three-dimensional local trees. (Figure 3.) And so on, to arbitrary dimension. This hierarchy does not collapse; there is no way of embedding the d-dimensional structures in the structures of any lower dimensionality without losing some of their structural properties. One way of showing this comes as a corollary of our characterization of Weir's control language hierarchy: there are sets of strings definable as the string yield of a set of structures at dimension d that are not definable with structures of any strictly smaller dimension.

2.1 Multi-Dimensional Tree Domains

In general, we will refer to d-dimensional trees domains as Td and the set of all d-dimensional tree domains as $\mathbb{T}^d$. We formalize these structures as a

generalization of Gorn's *Tree-Domains* [15, 16]. These are usually presented as sets of sequences of natural numbers (subsets of $\mathbb{N}^*$) interpreted as node addresses, with the root at address ε and the children of a node at address w at addresses $w \cdot 0, w \cdot 1, \ldots$ in left-to-right order. To be well-formed, then, a tree domain must be downward closed wrt concatenation (all prefixes of any sequence in the domain must also be present) and *left-sibling closed* in the sense that if $w \cdot i$ occurs in the tree domain then so does $w \cdot j$ for all $j < i$.

In these terms, a string domain is just an initial segment of $\mathbb{N}$ (a set of natural numbers downward closed wrt $<$) and a node address in a tree domain is a sequence of string addresses. In fact, the address of a node in a tree domain can be understood to be the sequence of addresses in the string yields of local trees of the points that one visits in following the path from the root to that node. We follow this pattern in generalizing to higher dimensions: an address of a node in a Td is a sequence of T$(d-1)$ addresses—the addresses in the yields of the local structures of the points one visits in following the path from the root to that node.

Thus, at dimension two and above addresses are sequences. To generalize downward, we represent the natural numbers as unary numerals, with 0 represented as ε and n represented as 1^n. Hence, an address in a string domain is a sequence of '1's, an address in an ordinary (2-dimensional) tree domain is a sequence of sequences of '1's, and, in general, addresses in d-dimensional tree domains are d^{th}-order sequences of '1's.

We begin, then, by making precise this notion of higher-order sequences.

DEFINITION 2.1 (Higher-order sequences of '1's)

- $^0 1 \overset{\text{def}}{=} \{1\}$.
- $^{n+1} 1$ is the smallest set satisfying:
  - $\langle \, \rangle \in {}^{n+1} 1$.
  - If $\langle x_1, \ldots, x_l \rangle \in {}^{n+1} 1$ and $y \in {}^n 1$, then $\langle x_1, \ldots, x_l, y \rangle \in {}^{n+1} 1$.

For example, $\langle \langle \langle 1 \rangle \rangle, \langle \langle 1 \rangle, \langle 1 \rangle \rangle, \langle \langle 1, 1 \rangle \rangle \rangle$ is a third-order sequence consisting of three second order sequences consisting, respectively, of a single first-order sequence, two first-order sequences and a single first-order sequence.

We represent the empty sequence (of any order) $\langle \, \rangle$ as ε and represent sequences of '1's without punctuation: $111 = \langle 1, 1, 1 \rangle$. Moreover, in order to simplify the notation, we will often exploit the intended interpretation of sequences of '1's as unary numerals and represent 1^n with the decimal numeral n.

Care must be taken to not confuse sequences of differing order and, in particular, to distinguish concatenation at distinct levels. Thus $\langle 1 \rangle \cdot \langle 11 \rangle = \langle 1, 11 \rangle$, not $\langle 111 \rangle$. This is complicated, slightly, by the fact that the empty sequence ε is common to all orders. There will be no ambiguity, however, because concatenation, henceforth, is defined only for sequences of the same order. So, for

instance, $\langle 11 \rangle \cdot 1$ is undefined and $\langle 11 \rangle \cdot \varepsilon = \langle 11 \rangle$ not $\langle 11, \varepsilon \rangle$ (which is $\langle 11 \rangle \cdot \langle \varepsilon \rangle$).

Definition 2.2 (Multi-Dimensional Tree Domains)

0-dimensional tree domains (point domains) are either empty or trivial:

$$\mathbb{T}^0 \overset{\text{def}}{=} \{\emptyset, \{1\}\}.$$

$(d + 1)$-dimensional tree domains are sets of $(d + 1)^{\text{st}}$-order sequences of '1's which are *heriditarily downward closed*:

$T \in \mathbb{T}^{d+1} \overset{\text{def}}{\Longleftrightarrow}$

- $T \subseteq \,^{d+1}1$,
- $(\forall s, t \in \,^{d+1}1)[s \cdot t \in T \Rightarrow s \in T]$,
- $(\forall s \in \,^{d+1}1)[\{w \in \,^{d}1 \mid s \cdot \langle w \rangle \in T\} \in \mathbb{T}^d]$.

The *leaves* of a d-dimensional tree domain are those points at addresses that are not properly extended by any other address in the tree domain. This is equivalent to being maximal wrt the d-dimensional successor relation. The *depth* of a tree domain is the length of the longest path of d-dimensional successors from the root to a leaf, which is to say, the length of the longest top level sequence it includes.

For any $s \in T$, a Td, $d > 0$, the *child structure* of s is the $\mathrm{T}(d-1)$ yield of the local Td rooted at s in T:

$$T_s = \{w \in \,^{d-1}1 \mid s \cdot \langle w \rangle \in T\}.$$

We will refer to $\{T_s \mid s \in T\}$ as the set of child structures of T and to the set of its child structures at any dimension (the child structures plus their child structures, etc.) as its set of *component structures* of the Td.

The *branching factor* of a Td *at a given dimension* $0 < i \le d$ is one plus the maximum depth of the component $\mathrm{T}(i-1)$ it contains. The *overall branching factor* is the maximum of its branching factor at all dimensions greater than 0. 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 if its branching factor is *no greater than n.*

2.2 Labeled Multi-Dimensional Trees

Usually, we will decorate the points in a tree domain with labels drawn from some alphabet Σ.

Definition 2.3 (Σ-Labeled Td)

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 $\mathbb{T}^{n,d}_\Sigma$.

Our goal is to define languages as the string yields of those sets of Σ-labeled
Td that satisfy logical formulae defining their structural properties. For these
purposes we model the n-branching Td as "initial segments" of T_n^d the relational
structure with T_n^d, the *complete n-branching Td* (i.e., the infinite Td in which
every point has a child structure that has depth $n-1$ in all its dimensions) as
the domain, along with relations for each of the d successor functions:

$$\mathsf{T}_n^d \stackrel{\mathrm{def}}{=} \langle T_n^d, \lhd_i \rangle_{1 \leq i \leq d}$$

where $x \lhd_i y$ iff x is the immediate predecessor of y in the ith-dimension.

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 con-
nectives, 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

$$\mathsf{T}_n^d \models \varphi\,[\boldsymbol{s}].$$

DEFINITION 2.4 (wSnTd)
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.

LEMMA 2.5
wS1T1 is equivalent to wS1S in the sense of interinterpretability, as is wS1Td
for all d. wSnT2 is interinterpretable with wSnS for all $2 \leq n \leq \omega$.

2.4   Definability in wSnTd

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

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

THEOREM 2.6
A set of strings over Σ is definable in wS1T1 iff it is definable in wS1S. A set
of Σ-labeled trees is definable in wSnT2 iff it is definable in wSnS.

2.5 Local and Recognizable Sets

The great strength of this hierarchy of structures is the uniformity of its definition. This allows easy generalization of known mechanisms and results at one dimension to structures of any dimension. For instance, we can define notions of grammars and automata over objects of arbitrary dimension by generalizing the familiar mechanisms over the first- and second-dimensional structures: finite-state automata over strings (and, subsequently, trees) and context-free grammars (over strings, but generating trees). We interpret these declaratively, as licensing sets of labeled structures. From this perspective, the automata turn out to be an extension of the grammars.

DEFINITION 2.7 (Td Grammar)
A *Td grammar* over an alphabet Σ is a finite set of Σ-labeled local Td.

As the trees in the grammar are local, they consist of a labeled root along with a yield: a labeled T$(d-1)$. We can interpret this as a production by taking it to license the rewriting of the symbol labeling the root as the structure of the yield. At this point, though, we will restrict our attention to the derivation structures—the labeled Td recording a derivation employing productions of this sort—rather than the labeled T$(d-1)$ such a derivation might yield. From this perspective the productions simply license local fragments of the derivation structures: a labeled Td is licensed by a Td grammar iff it is constructed from the local Td of the grammar.

When we wish to emphasize the interpretation of the local trees as productions, we will represent them as pairs in $\Sigma \times \mathbb{T}_{\Sigma}^{d-1}$, where the left member of the pair is the label of the root of the local tree and the right member is its yield structure.

DEFINITION 2.8 (Local Set)
The set of labeled Td licensed by a grammar $\mathcal{G} \subset \mathbb{T}_{\Sigma}^{d}$ relative to a set of *initial symbols* $\Sigma_0 \subseteq \Sigma$ (denoted $\mathcal{G}(\Sigma_0)$) is the set of all Σ-labeled Td with root labeled with a symbol in Σ_0 in which every local Td is included in $\mathcal{G}$.

A set of Σ-labeled Td is a *local set* iff it is $\mathcal{G}(\Sigma_0)$ for some Td grammar $\mathcal{G}$ over Σ and some $\Sigma_0 \subseteq \Sigma$.

(More complete definitions the local and recognizable sets can be found in [34, 36, 37].)

The T2 grammars are a mild generalization of ordinary Context-Free Grammars: we allow multiple start symbols and do not distinguish terminals and non-terminals in the standard way—any symbol may label the root of a local tree in the grammar and may, therefore, be expanded. It is possible, however, to distinguish a set of terminal symbols. Every local tree in the licensed tree must occur in the grammar. This includes the trivial local trees (trees with empty yields) occurring at the leaves of the tree. Hence every leaf must be

explicitly licensed by a trivial local tree in the grammar and the set of symbols decorating these trivial local trees are the symbols licensed to be terminals. From this perspective, the generalization simply allows terminals optionally to be expanded. These generalizations do not, of course, affect the string languages the local sets of trees yield, which are just CFLs.

The local sets of strings are the *strict 2-locally testable languages*, a weak subclass of the regular languages.

The Td automata are extensions of the grammars in which the structural licensing is based not (only) on the explicit labels of the nodes but (also) on a finite set of states.

DEFINITION 2.9 (Td Automaton)
A Td automaton over an alphabet Σ and a finite set of states Q is a finite set of pairs from the product of Σ and the set of Q-labeled local Td.

We can interpret an automaton as a grammar over Q (the second components of the pairs) along with a mapping from the local trees in that grammar to symbols in Σ. The Σ-labeled Td licensed by such an automaton are images of the Q-labeled Td licensed by the grammar in which the label of the root of each local Td has been replaced with a symbol in Σ associated with that local Td in the automaton.

DEFINITION 2.10 (Recognizable Set)
A Σ-labeled Td, $\mathcal{T} = \langle T, \tau \rangle$, is licensed by a T$d$ automaton $\mathcal{A} \subset \Sigma \times \mathbb{T}_Q^d$ relative to a set of *initial states* $Q_0 \subseteq Q$ iff there is a Q-labeled Td, $\mathcal{T}' = \langle T, \tau' \rangle$ in which the root is labeled with a symbol in Q_0 and in which every local Td is included as the right component of a pair in $\mathcal{A}$ and where, for each $w \in T$, $\tau(w) = \sigma$ only if there is a pair in $\mathcal{A}$ associating σ with the local Td rooted at w in $\mathcal{T}'$. The set of Σ-labeled Td licensed in this way is denoted $\mathcal{A}(Q_0)$.

A set of Σ-labeled Td is a *recognizable set* iff it is $\mathcal{A}(Q_0)$ for some Td automaton over Σ and Q and some $Q_0 \subseteq Q$.

The Q-labeled tree, $\mathcal{T}'$, is referred to as a *run* of the automaton. The automata can be thought of as building parallel structures simultaneously, with the structural configuration determined in the run and the external form determined in its Σ-labeled image.

The one-dimensional automata are the ordinary non-deterministic finite-state string automata, hence the recognizable sets of strings are just the regular sets of strings. The two-dimensional automata are standard non-deterministic finite-state tree-automata [14].

2.6 Definability and Recognizability

The recognizable sets at each dimension are characterized by concatenation of local structures—in this sense they are all *regular*—and, as a consequence, they

all exhibit similar properties. The uniformity of their definition makes it easy to generalize proofs of a property of recognizable sets at some specific dimension to proofs of that property at arbitrary dimensions—in essence, the dimension becomes a parameter of the proof, determining the type of structures manipulated by the constructions but playing no essential role in the constructions themselves.

A simple example is the relationship between the local and recognizable sets due originally, for strings, to Chomsky and Shützenberger [5] and, for trees, to Thatcher [43].

THEOREM 2.11
A set of Σ-labeled Td is recognizable iff it is a projection of a local set.

Here a projection is an arbitrary mapping (typically many-to-one) from one alphabet to another. For the forward direction, one constructs, from an arbitrary automaton over Σ and Q, a corresponding grammar over $\Sigma \times Q$ which licenses sets of Td in which the Q-labeled Td obtained via the right projection is a run of the automaton licensing the Σ-labeled Td obtained via the left projection. For the other direction one constructs an automaton with the original Σ as the set of states and the image of Σ under the projection as the label alphabet: given an arbitrary grammar $\mathcal{G}$ over Σ and projection $\pi : \Sigma \to \Sigma'$, for each local tree $\mathcal{T} \in \mathcal{G}$, the automaton includes the pair $\langle \pi(\sigma), \mathcal{T} \rangle$, where σ is the label of the root of $\mathcal{T}$.

Similar uniform proofs can be obtained for closure of the class of recognizable sets at each dimension under projection, cylindrification and Boolean operations, for closure under determinization (in a "bottom-up" sense), and for decidability of emptiness. (Additional examples can be found in [32, 34, 37].) Together, these properties allow us to lift the descriptive characterizations of Büchi, Elgot, Doner, and Thatcher and Wright to descriptive characterizations of the class of recognizable sets of arbitrary dimension.

THEOREM 2.12
A set of finite Σ-labeled Td is recognizable iff it is definable in $wSnTd$ for some $n < \omega$.

This gives us our initial descriptive characterizations of language-theoretic complexity classes. The recognizable sets of strings are just the regular (string) languages. Hence, definability in $wSnT1$ characterizes the regular languages. Similarly, the string yields of the recognizable sets of trees are the context-free (string) languages. (This is a consequence of Theorem 2.11.) Hence, definability (as yields) in $wSnT2$ characterizes the context-free languages.

COROLLARY 2.13 (Büchi'60, Elgot'61)
A set of (finite) strings is regular iff it is definable in $wSnT1$.

COROLLARY 2.14 (Doner'70, Thatcher and Wright'68)
A set of strings is context-free iff it is the yield of a set of finite trees definable in wSnT2.

3 wSnT2, $n < \omega$—GB

The first linguistic theory we will look at is Government and Binding Theory [6, 7]. Until the emergence of Minimalist Grammars [8] this was the dominant theory in the Principles and Parameters tradition. It consists of a loose and dynamically changing collection of universal principles constraining the form of phrase markers. These principles are tailored to particular languages by the settings of a small set of discrete-valued parameters. Hence it is very much a constraint-based theory—the range of licensed analyses is determined by the conjunction of structural properties, not by mechanisms for deriving them. As such, the complexities either of individual principles or of the overall theories are quite difficult to establish [2] and computational methods based on GB theories are, for the most part, speculative [12, 18].

It turns out that most, but not all, aspects of standard GB theories of syntax can be captured within wSnT2, and the distinction between what can and cannot be defined provides a characterization, in GB terms, of what it means to be Context-free. We can only provide a glimpse of these results here. For details see [33].

3.1 Definability

Typically, GB principles are stated in terms of *features*—finitely valued attributes of the constituents of an utterance—and a small suite of basic structural relationships between those constituents. Our general approach is to define each feature and relationship separately—effectively adding them to the signature of our language. We can then employ them freely in defining the properties based on them.

We can model features with a distinct monadic predicate for each possible feature value. This will be satisfied at a node iff the sub-tree rooted at that node represents a constituent bearing that feature with that value. As we can treat these additional monadic predicates simply as existentially bound second-order variables we can add any finite number of them without extending the expressive power of our language.

For the structural relationships we must be more careful. As these are typically non-monadic we cannot simply extend our signature to include predicates for them. As we shall see, adjoining even a single uninterpreted dyadic predicate to the signature of wSnT2 properly extends its descriptive power. Rather, any non-monadic predicates we add must be *explicitly* defined: it must be

possible to eliminate all occurrences of the predicates via a process of syntactic substitution by their definitions. For the most part, what this means is that, while non-monadic predicates can (and will) be defined in terms of other non-monadic predicates in addition to the predicates of the signature and any defined monadic predicates, there must be no circularity in their collective definition. It should be emphasized that the restriction to explicitly defined predicates does *not* extend to monadic predicates. Since monadic first-order implicit and inductive definitions can be captured as explicit monadic second-order definitions we have considerable freedom in defining monadic predicates.

Definitions within GB theories, while tending to be somewhat loose in formalization, follow, for the most part, an axiomatic style. Hence the process of translating into wSnT2 is actually reasonably straightforward. As an example, consider the fundamental GB relationships of C-Command and Government. There are a variety of minor variations on the way these are defined, but a common definition of C-Command has a category C-Commanding another iff neither category dominates the other and every branching category that dominates the first dominates the second. Government (in a highly-simplified form) can then defined in terms of C-Command: a category Governs another iff it C-Commands it and no Barrier intervenes.

$$
\begin{aligned}
\text{Branching}(x) &\equiv (\exists y, z)[x \lhd_2 y \land x \lhd_2 z \land y \not\approx z] \\
\text{C-Command}(x, y) &\equiv \neg x \lhd_2^* y \land \neg y \lhd_2^* x \land \\
&\quad (\forall z)[(z \lhd_2^* x \land \text{Branching}(z)) \to z \lhd_2^+ y] \\
\text{Governs}(x, y) &\equiv \text{C-Commands}(x, y) \land \\
&\quad \neg(\exists z)[\text{Barrier}(z) \land z \lhd_2^+ y \land \neg z \lhd_2^+ x]
\end{aligned}
$$

Here $\lhd_2^+$ and $\lhd_2^*$ are the transitive and (respectively) reflexive transitive closures of $\lhd_2$. These are explicitly monadic second-order definable via the definition of a *Branch*—a set of nodes upwards closed wrt and linearly ordered by $\lhd_2$:

$$
\begin{aligned}
\text{Branch}(X) \equiv\ & (\forall x, y)[(X(x) \land y \lhd_2 x) \to X(y)] \land \\
& \text{—if } x \in X \text{ has a parent } y \text{ then } y \text{ is also in } X \\
& (\forall x, y, z)[(X(x) \land X(y) \land X(z) \land x \lhd_2 y \land x \lhd_2 z) \to y \approx z] \\
& \text{—if } x \in X \text{ then } x \text{ has at most one child in } X.
\end{aligned}
$$

Then two nodes are related by $\lhd_2^*$ iff every Branch containing the second also contains the first:

$$
\begin{aligned}
x \lhd_2^* y \equiv\ & (\forall X)(\text{Branch}(X) \land X(y)) \to X(x) \\
& \text{—} x \textbf{ \textit{dominates} } y \textbf{ iff every Branch including } y \\
& \quad \textbf{includes } x \textbf{ as well.} \\
x \lhd_2^+ y \equiv\ & x \lhd_2^* y \land x \not\approx y \\
& \text{—} x \textbf{ \textit{properly dominates} } y \textbf{ iff it dominates but is} \\
& \quad \textbf{not equal to } y.
\end{aligned}
$$

CP
NP$_i$ C̄
Who   C   IP
I$_j$ C NP   Ī
do   ∅ you t$_j$ VP
V̇
V   CP
think t$_i$ C̄
C   IP
∅ t$_i$ Ī
I   VP
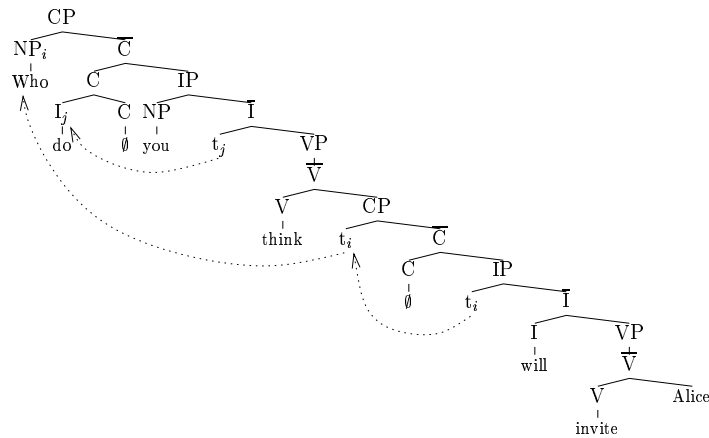will   V̇
V   Alice
invite

Fig. 4. Chains in GB.

Equality ($\approx$) and symmetric domination are equivalent.

Branching is then explicitly defined in these terms and C-Command is explicitly defined in terms of Branching. Governs is then defined in terms of C-Command and the monadic predicate Barriers, typically defined in terms of specific features, which we assume is defined elsewhere.

3.2   Non-definability

In many respects, the question of what aspects of GB can be defined within wSnTd is not as interesting as the question of what aspects cannot be defined. This boundary can be delimited by exploring the theory of *chains*—the mediator of movement in GB—and, in particular, the mechanism of *free-indexation* which is generally employed to identify the members of a chain.

A typical GB analysis of movement is illustrated in Figure 4. The *Who* has raised from its *base* position as the embedded subject (the specifier position of the embedded IP) to a *target* position at the front of the sentence (in the specifier position of the CP) passing through the specifier position of the embedded CP on the way. These three positions form a chain: a moved element along with a sequence of *traces* marking the positions it has moved from. This particular chain is referred to as an Ā-chain as it involves movement into a non-argument (Ā) position. A second chain marks the movement of the element carrying inflection (*do*) from the head of the IP to adjoin[1] at the (empty)

---

[1] There are three senses of the term *adjoin* that we will encounter. Here we refer to what is sometimes called *Chomsky Adjunction* in which a node (I here) is adjoined to another (C) by doubling the second and attaching the first as the sibling of the lower copy. A second sense will be encountered in Section 5:

head of the CP. Constraints on movement are represented as constraints on the form of these chains.

The difficulty in capturing these accounts in wSnTd is the mechanism generally employed to identify the members of the chain—free indexation—in which it is assumed that indices are assigned to elements of the tree randomly with constraints on chains and similar co-indexed structures filtering out the ill-formed assignments. The problem with free-indexation is that the indices, in essence, form an equivalence relation with unbounded index. Such a relationship is not definable in wSnTd which we can show by reduction from Lewis's *monadic second-order theory of the grid* [24].

The theory of the grid is the theory of the discrete first quadrant of the Cartesian plane with functions for immediate right- and above-successors. Lewis established undecidability of the MSO theory of the grid by reduction from the halting problem, capturing configurations of Turing Machines as horizontal sequences of labels and computations as the vertical evolution of these sequences. The theory of the grid is, in essence, the theory of the two branching tree modified by the property that the successors commute: the element above the right successor of a point is identical to that to the right of the element above it. If one adjoins an arbitrary equivalence relation to the signature of wS2T2 (indeed, as the property of being an equivalence relation is definable, any uninterpreted binary relation will do) the effect of the commutativity can be defined. Hence, the theory of the augmented structure is undecidable. If the adjoined relation were definable, it could be eliminated and, consequently, the theory of the augmented structure would be decidable by virtue of the decidability of the original theory. Thus, the adjoined relation is a strict extension—it cannot be defined within the original structure.

Evidently, then, if one is to capture a GB account of syntax one must do so without the use of indices (or, at least, without an unbounded supply of indices). Remarkably, more recent GB theories have tended to minimize and even eliminate the use of indices in favor of constraints on local relationships such as antecedent-government [28, 25]. One such theory is Rizzi's *Relativized Minimality* [28] which requires the antecedent-governor of a trace to be in the closest potential position for an antecedent-governor of the requisite type. This

---

the *tree adjoining* operation of TAGs. Finally, we also use *adjoin* in the standard model-theoretic sense of extending the signature of a class of models.

type or relationship is easy to capture in wSnTd:

$$\text{Abar-Antecedent-Governs}(x, y) \equiv$$
$$\neg\text{A-pos}(x) \wedge \text{C-Commands}(x, y) \wedge \text{F.Eq}(\underline{x, y})\wedge$$
—x **is a potential antecedent in an** $\overline{\text{A}}$**-position**
$$\neg(\exists z)[\text{Intervening-Barrier}(z, x, y)]\wedge$$
—**no barrier intervenes**
$$\neg(\exists z)[\text{Spec}(z) \wedge \neg\text{A-pos}(z)\wedge$$
$$\text{C-Commands}(z, x) \wedge \text{Intervenes}(z, x, y)]$$
—**minimality is respected**

Such relations can be extended with constraints on the co-occurrence of features, etc., to define *Link* relations—relations which must hold between adjacent elements of a chain—for each type of chain admitted by the theory. As these link relations are mutually exclusive, they can be combined into a single disjunctive relation Link(x, y) which must hold between adjacent elements of chains of any type. Chains, then, can be defined as, in effect, discrete linear orders (with end-points) wrt the Link relation:

$$\text{Chain}(X) \quad \equiv \quad (\exists!x)[X(x) \wedge \text{Target}(x)] \wedge (\exists!x)[X(x) \wedge \text{Base}(x)]\wedge$$
—X **contains exactly one Target and one Base**
$$(\forall x)[X(x) \wedge \neg\text{Target}(x) \rightarrow (\exists!y)[X(y) \wedge \text{Link}(y, x)]] \quad \wedge$$
—**All non-Target have a unique successor in** X
$$(\forall x)[X(x) \wedge \neg\text{Base}(x) \rightarrow (\exists!y)[X(y) \wedge \text{Link}(x, y)]] \quad \wedge$$
—**All non-Base have a unique antecedent in** X
$$(\forall x, y)[X(x) \wedge (\text{Link}(x, y) \vee \text{Link}(y, x)) \rightarrow X(y)]$$
—X **is closed wrt the Link relation**

The last constraint, requiring chains to be closed wrt the Link relation, when combined with the requirement that the ordering of the Link relation be linear, rules out the possibility of a single point participating in multiple chains.[2] In particular, this rules out structures in which a single element has moved from more than one base position, as in Figure 5. This requires, however, that we can always distinguish chains when they overlap and this proves to be the characteristic limitation of the approach.

3.3   *Context-Freeness in GB Theories*

By the characterization of Corollary 2.14, the string yields of the sets of phrase markers we can define in this way are all context-free languages. As it is

---

[2] There are GB accounts in which two chains of different types are joined by a common endpoint, but together they represent two phases of movement of a single element. We interpret these as a single chain with restrictions on the configurations in which the joining can occur falling out of the compatibility of the various link relations.
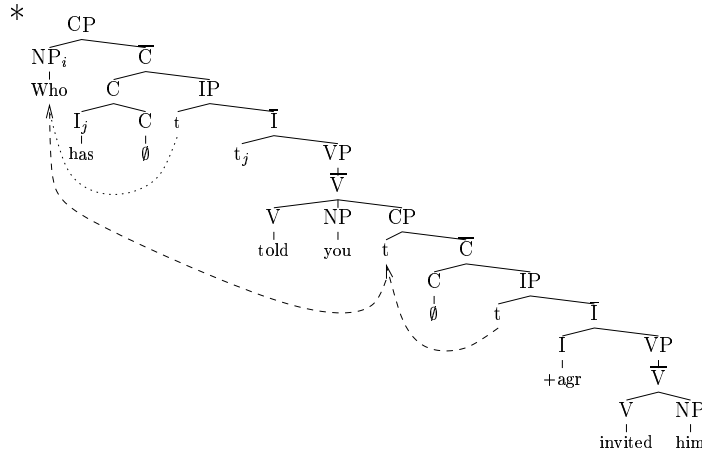
CP

*

Fig. 5. Conflated chains.

clear that there are human languages that are non-context-free [41, 17], the account must fail for some structures. A familiar example is the more-or-less standard account of head-raising in Dutch (Figure 6). As it turns out, these structures necessarily involve the potential overlap of unboundedly many chains. Consequently, a finite set of labels or features cannot serve to always distinguish those chains that overlap—no matter how we attempt to capture this account within wSnT2, there will always either be well-formed structures that we cannot license or ill-formed structures we do license.

In this way, the ability to fix *a priori* finite bounds on the number of overlapping chains serves as a diagnostic for context-freeness in GB terms:

- A language is "English-like" iff the number of chains which overlap at any single position in the tree is bounded by a constant.
- Standard GB accounts of English-like languages license strongly context-free languages.

4 Yields of Higher-Dimensional Trees

So far our mechanisms define only sets of labeled Td. Ultimately, we are interested in the string languages theses sets yield. For trees this is straightforward: the string yield is the set of nodes that are maximal wrt immediate domination ($\triangleleft_2$), ordered by immediate left-of ($\triangleleft_1$) extended in such a way that it respects the ordering of the interior nodes of the tree. (That is, if $x \triangleleft_1 y$ then all nodes dominated by x precede every node dominated by y.) For higher dimensional structures we will follow this same pattern, with the yield being the set of max-

FIG. 6. Head-raising in Dutch.

imal points wrt the successor in the major dimension, ordered by extensions of the successor relations in the minor dimensions. The yield of a Td will be a T$(d-1)$ with the string yield obtained by taking yields $d-1$ times.

There are two phases, then, to taking the yield. The first extends the orderings in the minor dimensions throughout the structure; the second restricts the resulting structure to maximal points. The crux of the process is the first phase; it, in essence, determines the way in which the yields of the child structures splice together. The difficulty is that, for dimensions higher than two, this is ambiguous. In Figure 7, for example, it is clear that B must dominate H in the yield. The question is which of I or J dominate F and G. Each choice
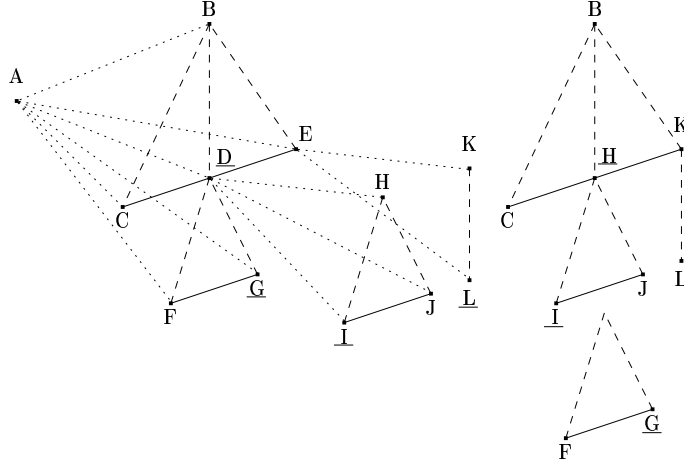
FIG. 7. Ambiguity in the yield of higher-dimensional structures.

reflects a different way of splicing the yields together.

We resolve this by building an explicit splicing into the structures. In each component local Ti we distinguish a point in the yield, the ith-*dimensional head* of that structure. In the yield of a each local Td, then, there is a sequence of $(d-1)$-dimensional heads extending from the root of the yield (a non-head) to a leaf. We refer to this sequence as the *primary spine* of the yield and refer to the leaf it includes as its *foot*. This foot designates the splicing point. In the figure, the 2nd-dimensional heads are indicated with underlines. Thus I is the foot of the $\{H, I, J\}$ tree and it (but not J) dominates F and G in the yield. (Hence, the two-dimensional primary spine of the yield of the full structure is $\langle B, H, I, G \rangle$.) In extending domination through the original structure, then, a node will inherit, from its ancestors in a dimension i, ancestor relationships in dimension $i - 1$ (so $D \triangleleft_3^+ J$ and $B \triangleleft_2^+ D$ implies $B \triangleleft_2^+ J$) but will only inherit descendant relationships if it (and all nodes on the path from the ancestor to the node) falls on the $(i - 1)^{\text{st}}$-dimensional primary spines of their component structures. (Thus, $D \triangleleft_3^+ I$ and $D \triangleleft_2^+ F$ implies $I \triangleleft_2^+ F$ but $D \triangleleft_3^+ J$ and $D \triangleleft_2^+ F$ does not imply $J \triangleleft_2^+ F$.)

There are a number of technical details involving the distribution of heads, the way domination at dimensions lower than $i - 1$ is extended and the way in which heads are determined in the result (for details see [37]) but, in the end, we get a class of structures in which the yield function corresponds, roughly, to restriction to maximal points:

DEFINITION 4.1
A Σ-*Labeled Headed* Td is a structure:

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

where T is a Td, $\vartriangleleft_i^+$ is the extended $\vartriangleleft_i$, R_i picks out the roots of the local Ti contained in T, H_i picks out the i-dimensional heads, and P_σ picks out the points labeled σ.

DEFINITION 4.2 (Yield)
The ith-*dimensional yield*, for each $1 \leq i < d$, of a Σ-labeled headed Td $\mathcal{T} = \langle T, \vartriangleleft_i^+, R_i, H_i, P_\sigma \rangle_{1 \leq i \leq d, \sigma \in \Sigma}$ is

$$\text{Yield}_d^i(\mathcal{T}) \stackrel{\text{def}}{=} \langle T^i, \vartriangleleft_j^+ \mid_{T^i}, R_j^i, H_j^i, P_\sigma \mid_{T^i} \rangle_{1 \leq j \leq i, \sigma \in \Sigma}$$

where T^i is the set of points in T that are maximal wrt $\vartriangleleft_j^+$ for all $i < j \leq d$ and the R_j^i and H_j^i are "images" (in a particular sense, see [37]) of R_j and H_j in T^i.

The string language licensed by a Td grammar or automaton, then, is $\text{Yield}_d^1(\mathbb{T})$, where $\mathbb{T}$ is the set of Σ-labeled Td it licenses.

5   wSnTd, $n < \omega$—**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 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 three-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*.

5.1   *Tree-Adjoining Grammars*

Briefly, a TAG consists of a set of *elementary trees* of two sorts: *initial trees* (α_1 in Figure 8) represent the minimal well-formed structures of the language, *auxiliary trees* ($\beta_1 \ldots \beta_3$ in the figure) represent the recursive structures—those expanding a constituent of a given category into a constituent of the same category. Each auxiliary tree, then, must have a node in its yield labeled with the same non-terminal as the root. This node is designated the *foot* of the tree.
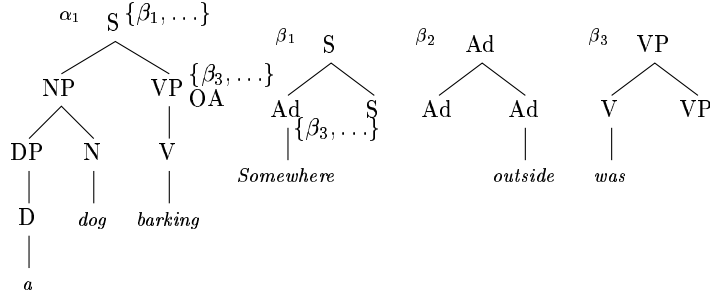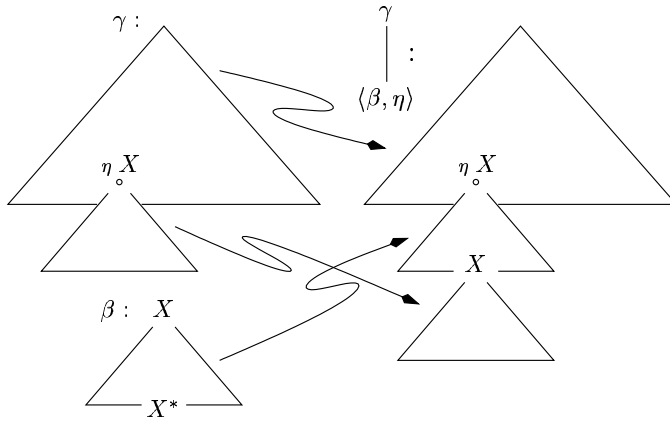
FIG. 8. TAG elementary trees.

FIG. 9. Adjunction in TAG.

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, to context-free rewriting in strings. It differs from the general context-free tree rewriting of Rounds ([38]) in that all children of the rewritten node are attached as the children a single node of the auxiliary tree with their order and multiplicity preserved.

In a *pure TAG* adjunction is controlled by the explicit labels of the nodes. An auxiliary tree may adjoin at a given node iff its root and foot bear the same label as that node. In practice, this is too weak and TAGs are augmented with

adjoining constraints associated with the nodes. These come three varieties: *selective adjoining (SA) constraints* designate the set of auxiliary trees which may adjoin at the node, *null adjoining (NA) constraints* forbid adjunction at the node (these are subsumed by empty SA constraints) and *obligatory adjoining (OA) constraints* require adjunction at the node. In the figure the SA constraints simply reflect the labeling constraints (although we might take the nodes with no SA constraint specified to have empty SA, and hence NA, constraints). The OA constraint on the VP node of α_1 reflects the fact that the gerundial form of *bark* requires an auxiliary (such as *was*) to be adjoined.

Given the mechanism of SA constraints, the restrictions on the labels of the trees is inessential. In fact, they are stipulations that reflect aspects of the linguistic content of standard TAG grammars (see [32]). We will generalize TAGs by relaxing these stipulations.

DEFINITION 5.1
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. Moreover, as we have done with the Td grammars, we do not strictly partition labels into terminals and non-terminals. Consequently, there is no reason to strictly partition the elementary trees either. We will take all trees to have a designated foot node, will permit any tree to adjoin wherever SA constraints permit and will simply identify the initial trees as a designated subset of the elementary trees.

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.
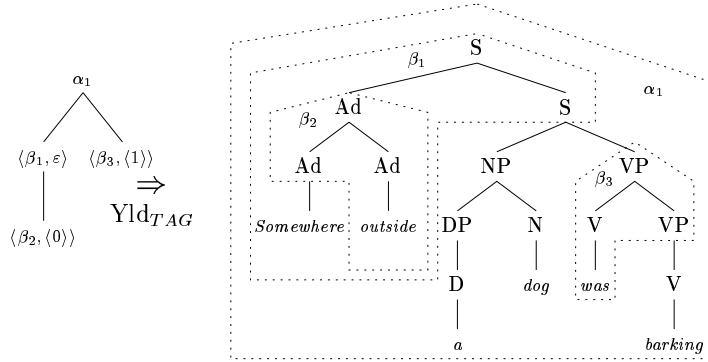
FIG. 10. Derivation and derived trees.

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

As Weir ([46]) has pointed out, these derivation trees are context-free. In the 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.

5.3 Equivalence of T3-Automata and Non-Strict TAGs

Equivalence of T3-automata and non-strict TAGs can be established by straightforward constructions—in essence, SA constraints and states have roughly equivalent power. Fuller details of these constructions can be found in [34, 35, 36, 37].

THEOREM 5.2
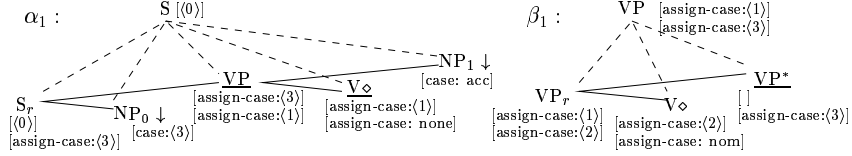A set of Σ-labeled trees is the yield of a recognizable set of Σ-labeled T3 iff it

$\alpha_1:$ S $[\langle 0\rangle]$ NP$_1 \downarrow$ [case: acc]

VP [assign-case:$\langle 3\rangle$] [assign-case:$\langle 1\rangle$]   V$_\diamond$ [assign-case:$\langle 1\rangle$] [assign-case: none]

S$_r$ $[\langle 0\rangle]$ [assign-case:$\langle 3\rangle$]   NP$_0 \downarrow$ [case:$\langle 3\rangle$]

$\beta_1:$ VP [assign-case:$\langle 1\rangle$] [assign-case:$\langle 3\rangle$]   VP* [ ] [assign-case:$\langle 3\rangle$]

VP$_r$ [assign-case:$\langle 1\rangle$] [assign-case:$\langle 2\rangle$]   V$_\diamond$ [assign-case:$\langle 2\rangle$] [assign-case: nom]

FIG. 11. Case assignment in XTAG (as local T3).

is generated by a non-strict TAG with adjoining constraints.

A T3 automaton can be constructed from a non-strict TAG by taking the set of states to be (roughly) the powerset of the set of elementary trees and building a set of local T3 for each elementary tree with the tree as the yield and the roots labeled with each state including the tree. For the other direction, we, in essence, convert each local T3 of the automaton into an elementary tree with the SA constraint associated with a node being the set of elementary trees formed from the yield of a T3 with root labeled with the same state as the node.

It is important to recognize just how direct these constructions are. In a very strong sense the T3 automaton and the TAG are just alternate presentations of the same object, with the local relationships of the automaton being expressed in the SA constraints of the TAG. 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.

COROLLARY 5.3
A set of Σ-labeled trees is generated by a non-strict TAG with adjoining constraints iff it is the yield of a wMSO definable set of finite Σ-labeled headed T3.

Thus, we can employ the wMSO language of Σ-labeled headed T3 as a declarative mechanism for defining TAGs, stating the syntactic constraints directly in logical terms and employing the constructions of the proofs of theorems 2.12 and 5.2 to compile them into TAGs. It should be noted that there is a significant practical problem with this approach in that the wMSO language is extremely expressive—it is possible to define infeasibly large TAGs with very small wMSO formulae. This is the same problem that exists in the application of the one- and two-dimensional characterizations in model-checking. Here, as in that application, the problem is one of avoiding infeasible definitions. Our belief, based in part on the experience of the model-checking application, is that, in many cases, this will be possible.

5.4   Applications

In linguistic applications SA constraints are usually encoded in the form of *feature structures*, a hierarchically structured generalization of the simple notion of feature employed in GB, in which the value of a feature may either be atomic or may be another feature structure. In addition, equivalence of the sub-structures rooted at certain points within the feature structures may be asserted with *re-entrancy tags*—essentially co-indexing those points. These can be seen in the trees forming the yields of the local T3 in Figure 11 (derived from the XTAG English grammar [49]) as '$\langle 1 \rangle$', etc. In general, feature structures may be recursive and satisfiability of a set of constraints on their form is not decidable.

In TAG usage, however, the feature-structures are required to form a finite set. Under these circumstances, we can capture them simply by taking each of the finitely many sequences of features that form paths through the admissible set of structures to be a monadic predicate, in essence labeling each point with the set of paths in the feature structure(s) assigned to it in the TAG. This also gives us a straightforward encoding of re-entrancy tags via path equations. If we let Feat denote the set of all paths occurring in the feature structures of the TAG then, for any $w, v \in$ Feat, we can assert equality of the value of w in the feature structure labeling a point x and the value of v in that labeling y with (letting ':' denote concatenation of feature sequences):

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

(Additional details can be found in [35].)

In feature structure based TAGs, in order to accommodate adjunction each node is associated with two feature structures, a *top* and a *bottom* feature structure. In the case that an auxiliary tree is adjoined at a node, the top feature structure of the node unifies with the top feature structure of the root of auxiliary tree and the bottom feature structure unifies with the bottom feature structure of its foot. If, on the other hand, no tree is adjoined at a node then the top and bottom feature structure of the node must unify with each other.

To see how this plays out linguistically, consider the elementary trees represented in the yields of Figure 11. The yield of α_1 is the initial tree for a gerundial form verb like *barking*. The yield of β_1 is an auxiliary tree for a auxiliary verb such as *was*. The obligatory adjunction of an auxiliary at the VP, in this account, is a consequence of the requirement that the subject NP must be marked with case. The verb, however, does not assign case and if the top and bottom feature structures of the initial tree are unified this requirement will be violated (via the re-entrancy tags $\langle 1 \rangle$ and $\langle 3 \rangle$). If the auxiliary tree is

adjoined at the VP, however, its sequence of assign-case features will intercept the connection between the subject and the main verb and the subject will receive nominative case from the auxiliary verb. Note that further adjunctions at the root of the auxiliary tree will intercept this sequence again. The result is that the subject receives case from the auxiliary that is most deeply embedded in the derivation, which will be the first in the sequence of auxiliaries in the string yield.

We can, in fact, adopt this account intact in a fully declarative logical definition as represented in the T3 of the figure, with feature structures now interpreted as sets of labels. In doing so, however, it becomes clear that many of the features involved are motivated by the derivational mechanism rather than by linguistic necessity. It seems highly unlikely, for instance, that the grammar writers seriously intend assign-case to be a linguistically significant feature of 'S's. Within the logical language, we can state relationships like that of the subject NP to the most deeply embedded auxiliary directly. The account of case-marking can become a component of the definition of α_1 by adding a constraint such as:

$$(\exists x, y)[vp \triangleleft_3^* x \wedge \mathrm{Max}_3(x) \wedge x \bar{\triangleleft}_2 y \wedge \langle \text{assign-case} \rangle(y) \wedge \langle \text{assign-case} = \text{case} \rangle(y, np_0)].$$

which says that the case feature of np_0 must agree with the assign-case feature of that child (in the second dimension—$\bar{\triangleleft}_2$ is the transitive closure of $\triangleleft_2$ restricted to local structures) that bears assign-case of a maximal descendent (in the third dimension) of vp (where that maximal descendent could possibly be vp itself).

This is only a first step, however. The identification of the case-marking verb in terms of its depth of embedding in the third dimension is still an artifact of the adjunction mechanism. We are not limited, however, to expressing our definitions in these terms. We might, for instance, define a notion of Government based on the structural relationships in a composite T3, something like:

$$\mathrm{Governs}(x, y) \equiv (\exists z)[z \triangleleft_3^+ x \wedge z \triangleleft_3 y \wedge$$
$$(\forall z')[(z \triangleleft_3^+ z' \wedge z' \triangleleft_3^* x) \rightarrow \neg \mathrm{Initial}(z')]].$$

which states that a node governs every child of each node properly above it, up to the first Initial node. This has each node governing all nodes in its own local T3 as well as all nodes in any T3 it adjoins into (i.e., is concatenated to) and all nodes governed by those it governs.[3]

We can then state the theory of case marking in linguistically more natural terms: case is assigned by the *maximal* governing case assigner, i.e., that which

---

[3]This account is not meant to be taken as a serious proposal. It is just one way of capturing the XTAG account in more familiar terms.

is not itself governed by any other case assigner.

$$(\forall x, y)[(\langle \text{assign-case}\rangle(x) \wedge \text{Governs}(x, y) \wedge \langle \text{case}\rangle(y)\wedge$$
$$\neg(\exists z)[\langle \text{assign-case}\rangle(z) \wedge \text{Governs}(z, x)]) \rightarrow$$
$$\langle \text{assign-case} = \text{case}\rangle(x, y)].$$

Note that there is no need of distinct top and bottom feature structures and no need of sequences of re-entrancy tags passing features around the trees. As in the GB account, we develop a language of linguistically significant structural relations and then define our theory of syntax in those terms. It should be noted, though, that the intermediate features do not simply disappear. Rather, they are reinstantiated as states and eventually adjoining constraints during the process of translating the logical definition into automata and TAGs. In this way the logical language behaves exactly like a higher-level language. It lets us define the set of structures in terms that are independently meaningful hiding the details of the mechanism for recognizing them.

6   Unbounded Branching

The branching factor of the local and recognizable sets is bounded by virtue of the finiteness of the grammars and automata. This is inconvenient for some accounts of syntax, particularly "flat" accounts of the syntax of coordination in which an arbitrarily large collection of constituents of similar category are combined into a single constituent of the common category. The branching factor of the wMSO definable sets, on the other hand, is bounded only by the branching factor (n) of the underlying class of models. Thus, there is nothing forcing us to adopt a fixed finite bound on the branching. If we take our underlying model to be T_ω^d (the ω-branching Td) that is, if we work in wSωTd, we can define sets of finite structures in which, while the branching factor of each individual Td is finitely bounded, there is no finite bound on the branching in the set as a whole.[4]

There are two questions raised by this idea. First, since the automata can no longer recognize the sets directly it is not immediately clear that wSωTd is decidable. Moreover, there is a question of how complicated these sets might be. In particular, since the set of yields of the local Td occurring the Td in the set is now potentially infinite, it is not clear what limit there may be on the complexity of the languages these definable sets may yield.

In the two-dimensional case decidability of the theory of the ω-branching structure (even of the full MSO theory) was established by Rabin [27] and we can both generalize his crucial lemma to arbitrary dimensions and use it as a means of establishing bounds on the complexity of the languages of the yields of the local Td and hence the yield languages.

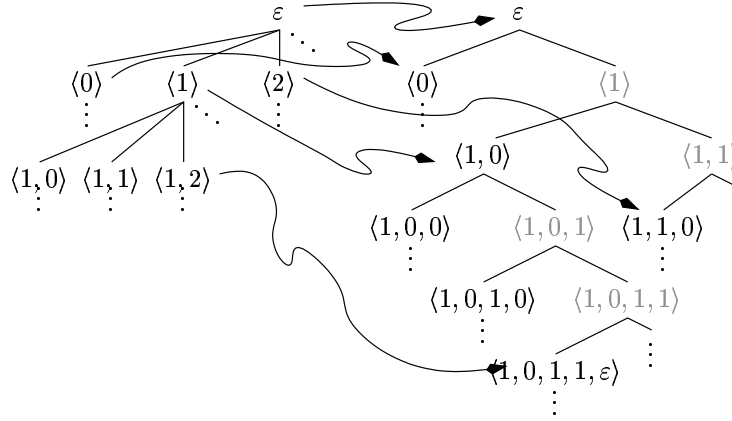[4] Note that this involves no change to the logical language, only to its interpretation.

FIG. 12. Embedding T^2_ω in T^2_2.

LEMMA 6.1 (from Rabin)
There is an effective translation $\varphi \mapsto \varphi'$ such that, for all $n \leq \omega$, $\varphi \in \mathrm{wS}n\mathrm{T}d$ iff $\varphi' \in \mathrm{wS2T}d$.

The idea of the translation is just that of the more-or-less standard conversion of an arbitrarily branching tree into a binary branching tree: each local tree maps to a right branching binary tree in which the children of the local tree are the leaves (left children) and the spine (the right children) are new nodes. (See Figure 12.) To find the image of the ith child of a node one follows i right successors from the image of that node and then a single left successor. Rabin's observation is that this embedding preserves domination and linear precedence and that the images of the nodes of the original tree form a MSO definable subset of the binary tree. Thus, given an arbitrary φ in the language of wSnT2, for any n including ω, we can relativize it to the image of T^2_n in T^2_2 to obtain the φ' of the lemma. It follows immediately that wSωT2 is decidable.

The generalization is somewhat messier, but reasonably straightforward (details are in [36]). Figure 13 illustrates the mapping of a single local T3. In the two-dimensional case the image of the n-branching tree was the set of roots of the string yields of the local trees in the two-branching tree. Here, the image is the set of roots of the T2 yields of the local T3 of the two-branching structure. (The dashed lines indicate the successor relationships of the original T3.) In general, the image is the set of roots of the $((d-1)$-dimensional) yields of the local Td. Again the image of the arbitrarily branching structure forms a definable subset of the two branching structure in which domination (in each dimension) is preserved. Thus we get decidability at all dimensions.
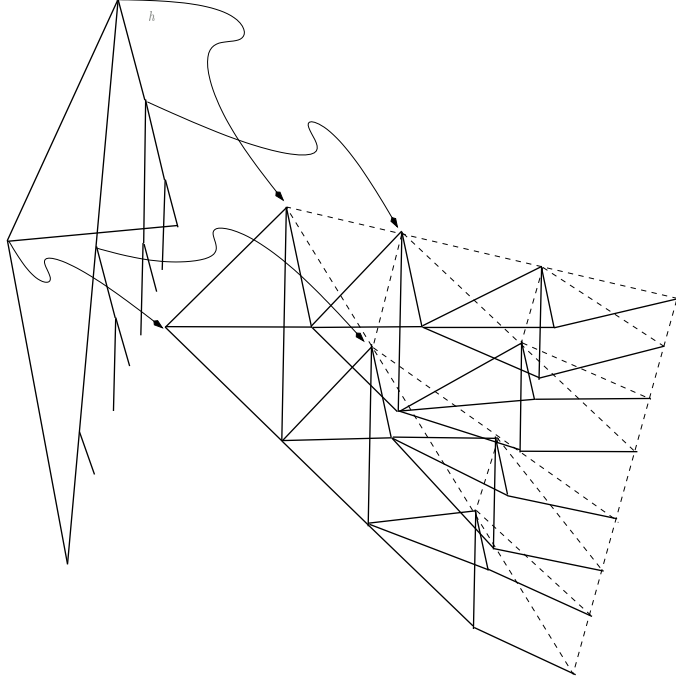
FIG. 13. Embedding T_ω^3 in T_2^3.

THEOREM 6.2
wSωTd is decidable for all $d < \omega$.

Moreover, since the mapping preserves domination and none of the non-image points in the two-branching structure are maximal, the string yields of the two structures will be identical. This gives a generalization of Chomsky Normal Form.

LEMMA 6.3 (Two-Branching Normal Form)
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 2-branching Td.

So, just as in the two-dimensional case, the extension to arbitrary finite branching does not extend the class of string languages the sets of definable structures yield. It remains to determine the extent to which it extends the classes of definable structures themselves. In particular, we are interested in the complexity of the sets of structures which form the yields of the local structures expanding a given label. Here we can show that it is possible to extract, for each label, a $T(d-1)$ automaton from the embedded structures
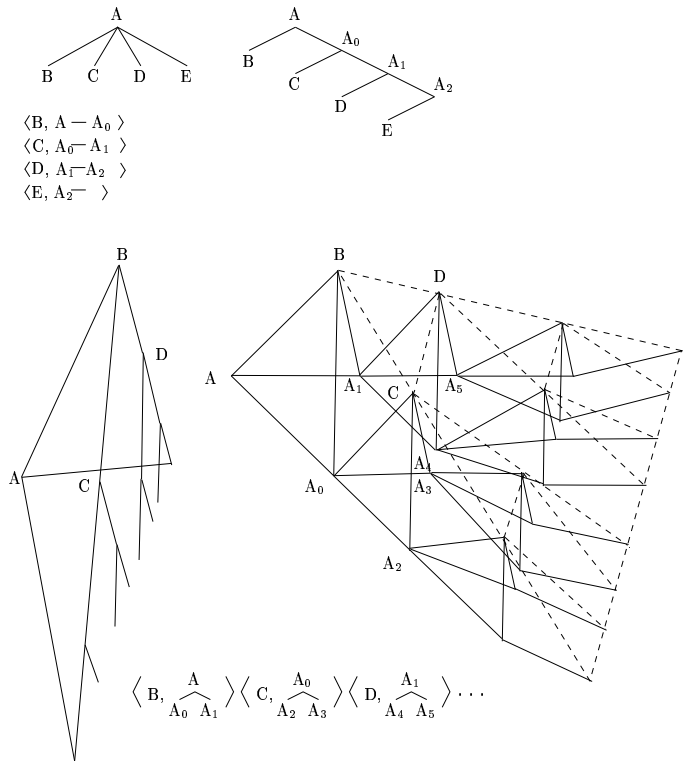
FIG. 14. Recognizability of $\mathrm{T}(d-1)$ yields of recognizable sets of $\mathrm{T}d$.

that recognizes the set of yields of local $\mathrm{T}d$ rooted at that label in the original set. Roughly, given the set of two-branching $\mathrm{T}d$ in which the local $\mathrm{T}d$ rooted at a given label are embedded, we form a $\mathrm{T}(d-1)$ automaton with the labels of the images of the yield as the label alphabet (B, C, ... in Figure 14) and the labels of the other nodes in the two-branching $\mathrm{T}d$ (its *spine*) as the states (the 'A's). The idea is that the spines of the two-branching $\mathrm{T}d$ form a run of the automaton which licenses the yield of the original $\mathrm{T}d$ as witnessed by the parallel structure formed by the image of that yield. (That is, we take the relationship between the root of the two-branching local $\mathrm{T}d$ and the root of its yield to be the projection mapping states to labels.) Hence the set of yields of the local $\mathrm{T}d$ rooted at that label in the original set of $\mathrm{T}d$ are recognizable.

This gives us the generalization of the grammars and automata that characterize the definable sets with unbounded branching.

DEFINITION 6.4 (Generalized Td Grammar/Automaton)
A *generalized Td grammar* over an alphabet Σ is a (potentially infinite) set of Σ-labeled local Td in which, for each $\sigma \in \Sigma$, the set of yields of the Td with root labeled σ is a generalized recognizable set of T$(d-1)$.

A *generalized Td automaton* over an alphabet Σ and a finite state set Q is a (potentially infinite) set of pairs from the product of Σ and the set of Q-labeled local Td in which the right projection of the set forms a generalized Td grammar over Q.

DEFINITION 6.5 (Generalized Local/Recognizable Set)
A set of Σ-labeled Td is a *generalized local set* iff it is $\mathcal{G}(\Sigma_0)$ for some generalized Td grammar $\mathcal{G}$ over Σ and some $\Sigma_0 \subseteq \Sigma$.

A set of Σ-labeled Td is a *generalized recognizable set* iff it is $\mathcal{A}(Q_0)$ for some generalized Td automaton $\mathcal{A}$ over Σ and Q and some $Q_0 \subseteq Q$.

Note that the one-dimensional case is degenerate; all T1 are unary-branching since the yields of the local structures are just points. In the two-dimensional case the grammars are, in essence, CFGs in which the set of productions is, itself, a regular set. Such grammars have been referred to as *extended* CFGs by Thatcher [43] and as *hypergrammars* by Langendoen [23]. As we shall see, there is a strong parallel between the sets recognized by generalized tree-automata and the sets of trees licensed by GPSG grammars. Our preferred terminology emphasizes this connection.

THEOREM 6.6
A set of Σ-labeled Td is definable in wSωTd iff it is a generalized recognizable set.

7   wSωT2—GPSG

Generalized Phrase-Structure Grammar (GPSG) [13] is a non-transformational account of syntax based on a number of generalizations of CFGs. To begin with, categories (labels) are not atomic but, rather, are drawn from a (finite) set of non-recursive feature structures. Secondly, productions are factored into two components: ID rules, which determine constituency but leave left-to-right order of the yield unspecified, and LP rules, which determine that order. The elements of these rules may be underspecified—they may refer to classes of categories (feature-structures) by specifying only the features that determine the class. Additionally, the individual elements in the yield of the ID rules may be iterated with a Kleene star. Thus, the set of ID rules is not necessarily finite but is regular. The interpretation of the ID and LP rules is precisely the same as the notion of licensing we use: a tree is licensed by the grammar iff each of its local trees is licensed by the rules. Finally, this system of licensing trees via constraints is extended with an extensive system of interacting restrictions,

NP

NP[CONJ NIL] NP[CONJ NIL] NP[CONJ *and*]

$\overline{\text{N}}$ $\overline{\text{N}}$ [SUBCAT *and*] $\overline{\text{N}}$
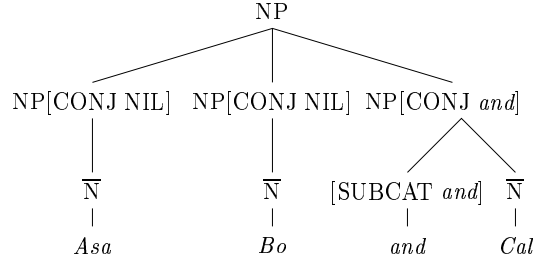
Asa *Bo* *and* *Cal*

FIG. 15. Iterated coordination in GPSG.

principles and constraints on the distribution of features in local trees and the set of ID rules is closed under a set of meta-rules that capture syntactic alterations such as question formation, passivization, topicalization, coordination, etc.

To a remarkable degree, these generalizations are consonant with the nature of definitions within $wS\omega T2$. The ID rules are all directly stateable in terms of $\lhd_2$, the LP rules in terms of $\lhd_1$. The iterated categories can be handled naturally using quantification; working on a foundation of T^2_ω permits us to license the unbounded branching of the corresponding sets of local trees. We have already seen how to handle finite feature structures by interpreting their paths as monadic predicates. The effect of the meta-rules can easily be accommodated by, in essence, "multiplying them out" in defining the ID rules. Finally, the effect of restrictions, principles and constraints can be captured as well, although some of them (FSDs in particular) involve definitions of some subtlety.[5] For details see [31, 30].

As an example, we will look at the GPSG account of coordination. Arbitrarily branching coordination (in contrast to the binary coordination of conjunctions like *but*) is licensed by the following rules:

$$X \longrightarrow H[\text{CONJ}_{\alpha_0}], H[\text{CONJ}_{\alpha_1}]^+$$
$$\text{where } \langle \alpha_0, \alpha_1 \rangle \in \{ \langle and, \text{NIL} \rangle, \langle \text{NIL}, and \rangle, \langle neither, nor \rangle, \ldots \}$$

$$X[\text{CONJ NIL}] \longrightarrow H$$
$$X[\text{CONJ } \alpha] \longrightarrow [\text{SUBCAT } \alpha], H$$

$$[\text{CONJ}_{\alpha_0}] \prec [\text{CONJ}_{\alpha_0}]$$
$$\text{where } \alpha_0 \in \{ both, either, neither, \text{NIL} \}$$
$$\text{and } \alpha_1 \in \{ and, but, nor, or \}$$

$$[\text{SUBCAT}] \prec [\sim \text{SUBCAT}]$$

The first of these is an ID rule schema (the *iterating coordination schema*) that

---

says that a member of any category (X is radically underspecified—it unifies with every category) can consist of two or more constituents, each marked as a head (H), one of which has a value α_0 for the CONJ feature and the rest of which have value α_1 for CONJ, where α_0 and α_1 are drawn from the specified set of pairs. The fact the constituents are marked as heads forces them, by the Head Feature Convention, to agree with their containing constituent on certain features, thus coordinated VPs form a VP, etc.

The second two ID rules simply realize the coordinating conjunctions. Categories of type [SUBCAT x] are lexical—they are expressed as the lexeme 'x'.

Finally, the last two items are LP rules the first of which (a schema) gives the relative left-to-right ordering of the of the various coordinating conjunctions and the second of which states that members of lexical categories precede those of non-lexical categories.

This gives accounts like that of Figure 15 in which arbitrarily many XPs, along with their appropriate conjunctions, combine to form another XP. The fact that both $\langle and, \text{NIL} \rangle$ and $\langle \text{NIL}, and \rangle$ are licensed admits both *Asa Bo and Cal*, in which the iterated conjunction is NIL, and *Asa and Bo and Cal*, in which the single conjunction is NIL.

The translation to wSωT2 is nearly immediate. The ID rules are interpreted as local constraints. The full translation requires one such constraint to hold for every local tree.

$$
\begin{aligned}
\text{CS}^+(x, y_1) &\equiv H(y_1) \wedge \langle \text{CONJ}, \alpha_0 \rangle (y_1) \wedge (\exists z)[x \lhd_2 z \wedge z \not\approx y_1] \wedge \\
&\quad (\forall z)[x \lhd_2 z \rightarrow (z \approx y_1 \vee H(z) \wedge \langle \text{CONJ}, \alpha_1 \rangle (z))] \\
\text{CONJ}_{\text{NIL}}(x, y) &\equiv \langle \text{CONJ}, \text{NIL} \rangle (x) \wedge H(y) \wedge \\
&\quad (\forall z)[x \lhd_2 z \rightarrow z \approx y] \\
\text{CONJ}_{\alpha}(x, y_1, y_2) &\equiv \langle \text{CONJ}, \alpha \rangle (x) \wedge \langle \text{SUBCAT}, \alpha \rangle (y_1) \wedge H(y_2) \wedge \\
&\quad (\forall z)[x \lhd_2 z \rightarrow (z \approx y_1 \vee z \approx y_2)]
\end{aligned}
$$

The LP rules are universal constraints holding for all siblings:

$$
\begin{aligned}
(\forall y_1, y_2)[(\exists x)[x \lhd_2 y_1 \wedge x \lhd_2 y_2] \rightarrow \\
(\langle \text{CONJ}, \alpha_0 \rangle (y_1) \wedge \langle \text{CONJ}, \alpha_1 \rangle (y_2) \rightarrow \neg y_2 \lhd_1 y_1)]
\end{aligned}
$$

$$
\begin{aligned}
(\forall y_1, y_2)[((\exists x)[x \lhd_2 y_1 \wedge x \lhd_2 y_2] \rightarrow \\
(\langle \text{SUBCAT} \rangle (y_1) \wedge \neg \langle \text{SUBCAT} \rangle (y_2) \rightarrow \neg y_2 \lhd_1 y_1)]
\end{aligned}
$$

The intent, here, is to demonstrate the directness of the translation between the GPSG grammar and the wSωT2 axioms. To a large extent, this is a consequence of the similarity between the descriptive facilities of the two formalisms. The relationship between the two is very much like that of TAG and wS2T3. There are linguistic principles that are implicit in the standard grammars which must be stated explicitly in the model-theoretic approach and the model-theoretic approach generalizes the grammars slightly,[6] but, to a large

---

[6] For instance, in wSωT2 the ID rules can form arbitrary regular sets, whereas in GPSG only individual
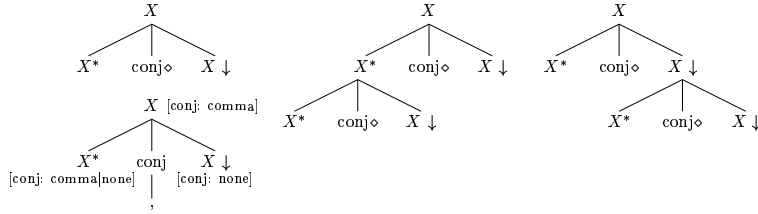
FIG. 16. Non-verb coordination in XTAG

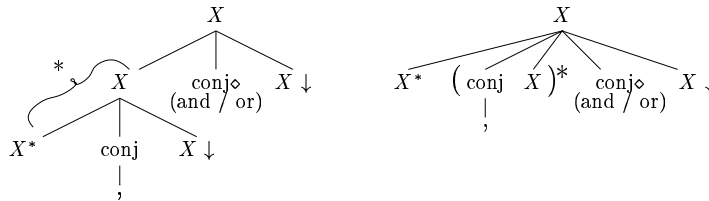
FIG. 17. Non-verb coordination in Generalized TAG

extent, they may be seen as alternative means of describing the same classes of languages.

8 wSωT3—Generalized Tree Adjoining Grammar

Since the step from wSnT2 to wSnT3 takes us, in essence, from CFGs to TAGs and the step from wSnT2 to wSωT2 takes us from CFGs to GPSG, we might say the step from wSnT3 to wSωT3 takes us from TAGs to *Generalized TAGs*. These are non-strict TAGs in which the set of elementary trees is required only to be recognizable or, equivalently, they can be taken to be generalized T3 automata.

Given GPSG's elegant account of coordination, this is the obvious place to look for potential usefulness of the added power of Generalized TAG. As it turns out, both the ability to 'flatten' the structure and the ability to factor out constraints of different sorts have the potential to simplify existing TAG accounts. A more detailed treatment can be found in [36].

8.1 *Ordinary Conjunction*

The XTAG grammar [49] treats coordination in two cases. Non-VP coordination is accomplished with a family of auxiliary trees anchored by conj adjoining

---

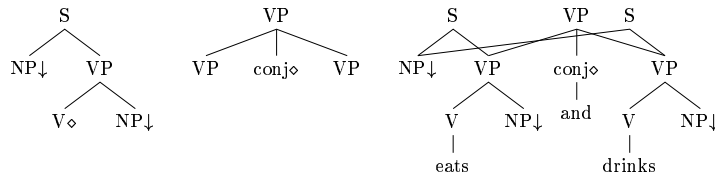constituents can be iterated.

FIG. 18. Predicative coordination in XTAG

[conj X] to the right of X (for X in Adj, A, P, PP, N, NP, Det, or S) (Figure 16) with additional trees handling multi-word conjunctions such as *either/or*. In the case of multiple conjunction this admits analyses with all possible nesting of the scopes. This is appropriate when the conjunctions are of the ordinary sort but it is a clear case of over-generation in the case of lists conjoined by commas. This is addressed by adding a 'conj' feature that prohibits adjunction of comma trees into conj trees of any other type.

As the set of trees formed by iterated binary conjunction is recognizable, one can accomplish the same end in Generalized TAG by treating the coordination as a single adjunction (Figure 17). This allows the idiosyncratic characteristics of the various conjunctions to be handled within single adjoined structures—not only limiting comma trees, for instance, to be left branching, but also accommodating unbounded multi-word iterated conjunctions like *neither/nor*. Since we are not limited to recognizable sets, but can admit unbounded branching in our auxiliary trees, we can push this flattening down to the second-dimension as well, adopting genuinely flat accounts of iterated conjunction. Moreover, moving to the signature of wSωT3 allows adoption of a GPSG-style factoring of constituency and precedence constraints. Thus one can, if one is so inclined, import the GPSG account of the syntax of coordination directly into Generalized TAG.

8.2   *Predicative Coordination*

VP coordination, such as gapping and right-node-raising constructions, have always been problematic in TAG in that, while the linguistic principles that motivate TAG require predicates and their arguments to occur within the same elementary structure, the constructions require arguments of multiple verbs to be realized by a single constituent. Joshi and Schabes [20] proposed to handle this with a mechanism that merges trees, collapsing their common structure (Figure 18).   Sarkar and Joshi [39] refine this by moving to DAGs for the derivation structure. This leaves open the question of whether to merge the derived trees or not. If one insists on traditional single-rooted trees the derived structures can be collapsed. On the other hand, for many purposes the deriva-

$$(\forall x)[(\mathrm{Verb}(x) \wedge \mathrm{Trans}(x))$$
$$\rightarrow ((\exists y)[\mathrm{Subj}(y,x)]$$
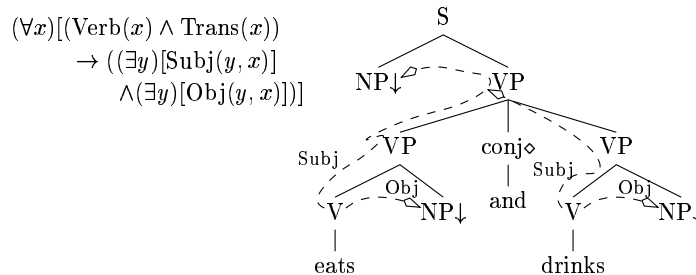$$\wedge (\exists y)[\mathrm{Obj}(y,x)])]$$

FIG. 19. Predicative coordination in Generalized TAG

tion tree itself is of more interest than the structure it describes; the fact that the structure it describes may not be entirely tree-like may be ignored.

Part of the attraction of approaching TAG from a model-theoretic point of view is that it allows one to treat the various constraints that determine the configurations of the elementary structures independently. As we have shown in [35] this allows modularization of the grammar of the sort suggested by Vijay-Shanker and Schabes [45] and developed variously by Becker [1], Evans and Weir and others at Sussex [11, 42], Candito [4], and Xia, Palmer, Vijay-Shanker and Rosenweig. [48] In particular, a verb, rather than selecting a specific set of elementary trees, may be thought of as selecting a set of constraints—all verbs might require, for instance, a subject, while transitive verbs would require, in addition, a direct object and ditransitive verbs an indirect object as well. Limitations on the actual configurations in which these arguments might be realized would, presumably, be consequences of independently motivated constraints (Figure 19). The set of elementary trees selected by a lexical item in the traditional presentation of a LTAG (Lexicalized TAG) grammar is the set of all trees satisfying the constraints it selects in this presentation.

This allows both derivation and derived structures to be of the traditional sorts. Instead it admits what amounts to elementary trees in which arguments may be missing—although they will be present in the form of constraints on where the tree may occur. In practice, when w$S\omega$T3 axioms are translated into T3 automata, such constraints will show up in the states assigned to the nodes in the structures—equivalent to introducing features to realize them. Thus, again, we can adopt something very close, in spirit, to a GPSG-style account of VP coordination, but without requiring the grammar writer to multiply out the details of the way in which constraints are propagated.

9 Higher Dimensions—Weir's Control Language Hierarchy

So far, we have established that wMSO definability in one dimension characterizes regular languages, in two dimensions characterizes CFLs and in three dimensions characterizes TALs. It's easy to see, as well, that definability in zero dimensions characterizes the finite sets. What is left is to determine the complexity of the string languages definable in dimensions higher than three. Here again, it turns out that there is a known hierarchy of language classes that is characterized by definability in our hierarchy of structures.

9.1 Weir's Control Language Hierarchy

In [47] Weir defines a hierarchy of *control languages*[7] based on *Labeled Distinguished Grammars* (LDG). These are, in essence, sets of labeled productions in which each right-hand side has a distinguished element, its *head*. As in our headed two-dimensional structures, for each node in a derivation tree of a LDG there is a unique sequence of heads leading from that node to a leaf. We will refer to these paths as the *spines* of the tree. Each spine is associated with a *control word*: the sequence of labels of the productions that fall along the path. The set of all control words occurring in a set of derivation trees is referred to as its *control set*.

Weir generates his hierarchy by associating LDGs with restricted control sets.

Definition 9.1
Let G be a LDG and C a language over the labels of G. Then $L(G, C)$ is the set of string yields of the derivation trees of G with control sets contained in C.

Let $\mathcal{C}_1$ be CFL.

Let $\mathcal{C}_{i+1}$ be the class of all $L(G, C)$ for some LDG G and $C \in \mathcal{C}_i$.

The fact that this hierarchy is proper has been shown by Palis and Shende [26] by establishing pumping lemmas for each level of the hierarchy.

To get a sense of how this mechanism corresponds to Td grammars, consider the LDG of Figure 20. The control set of the derivation tree in the figure is $\{P_1 P_2, P_3 P_4, P_2\}$. Consider, then, the derivation trees of $L(G, C)$ where $C \in CFL$ (so $L(G, C) \in \mathcal{C}_2$). As C is a CFL, there is some CFG, say G', that generates it. Thus, each of the control words in the derivation trees of $L(G, C)$ is the yield of a derivation tree in G'. (See Figure 21.) The idea is to build a T3 in which the derivation trees of the control words form "vertical slices" through the structure extending above the spines of the derivation tree of G.

---

[7]These are a generalization of Khabbaz's control languages [22] which correspond to the linear version of Weir's control languages—those in which each derivation tree is associated with a single control word.
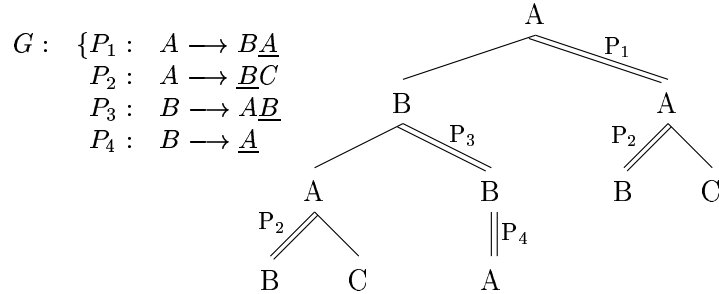
$$G: \quad \{P_1: \quad A \longrightarrow B\underline{A}$$
$$P_2: \quad A \longrightarrow \underline{B}C$$
$$P_3: \quad B \longrightarrow A\underline{B}$$
$$P_4: \quad B \longrightarrow \underline{A}$$
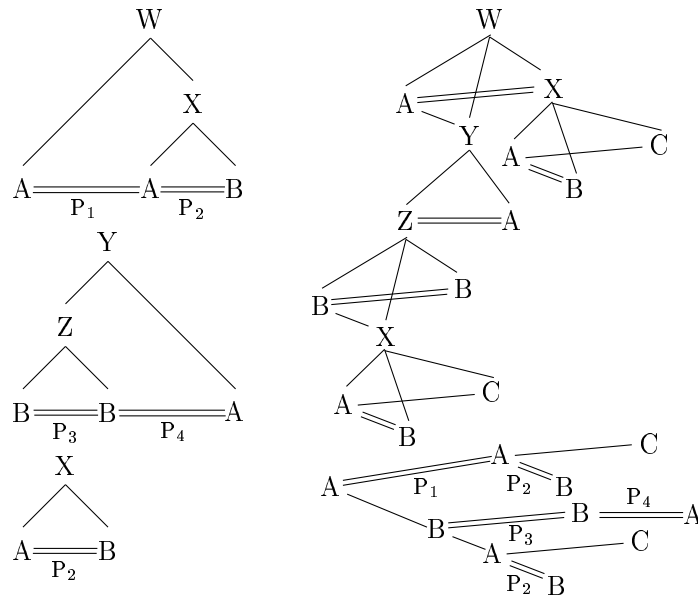
FIG. 20. A Labeled Distinguished Grammar.

FIG. 21. $\mathcal{C}_2$ derivation trees as T3.

Such a structure is given on the right-hand side of the figure. This turns out to always be possible and the construction generalizes easily to arbitrary $\mathcal{C}_i$ and arbitrary dimensions. Thus each language in $\mathcal{C}_i$ can be shown to be the string yield of a recognizable set of T$(i + 1)$.[8] (A detailed exposition of the constructions is in preparation [29].)

---

[8] The '+1' reflects the fact that Weir's hierarchy has the CFLs at level one while we have them at level two.

The proof of the opposite direction of the characterization involves a generalized notion of spine paralleling our notion of yields of arbitrary dimension. If $\mathbb{T}$ is a set of Σ-labeled headed $\mathrm{T}d$, for each $i \le d$ we define $\mathrm{Spine}_d^i(\mathbb{T})$ to be the set of i-dimensional "vertical slices" (again, in a particular sense) through the $\mathrm{T}d$ of $\mathbb{T}$. It can be shown that if $\mathbb{T}$ is recognizable then so is $\mathrm{Spine}_d^i(\mathbb{T})$ for each $i \le d$.

The key lemma establishes that these spine functions commute with the yield functions:

LEMMA 9.2

$$\mathrm{Spine}_j^i(\mathrm{Yield}_k^j(\mathcal{T})) = \mathrm{Yield}_{i-j+k}^i(\mathrm{Spine}_k^{i-j+k}(\mathcal{T})).$$

With this we get that the one-dimensional spines of the two-dimensional yields of a recognizable set of Σ-labeled headed $\mathrm{T}d$ are the one-dimensional yields of its $(d-1)$-dimensional spines. Since those $(d-1)$-dimensional spines are recognizable sets of $\mathrm{T}(d-1)$, we get, by induction on d, that their one-dimensional yields form a language in $\mathcal{C}_{d-2}$. Consequently, the one-dimensional yield of the set is in $\mathcal{C}_{d-1}$.

THEOREM 9.3
A string language is $\mathrm{Yield}_d^1(\mathbb{T})$ for some $\mathbb{T}$, a recognizable set of $\mathrm{T}d$, $d \ge 2$, iff it is in $\mathcal{C}_{d-1}$.

9.2 Applications

One potential linguistic application of the higher-dimensional structures provides a potential resolution to a well-known semantic difficulty with the standard TAG account of modifiers. This is a result of the fact that there are selectional constraints that restrict where modifiers adjoin. The locative PP *to the mailbox*, for instance, may modfy the verb *walk* but not the *feed*.

Asa walked her dog to the mailbox
⋆ Asa fed her dog to the mailbox

Here, there is no problem. The SA constraint associated with the VP can either permit or block adjunction of locative PPs. The problem occurs when there are multiple modifiers. The selectional restrictions hold no matter how many modifiers there are and in whatever order they are adjoined.

Asa walked her dog yesterday to the mailbox
⋆ Asa fed her dog yesterday to the mailbox

The problem is that in the standard TAG account, for this word order, the PP does not adjoin directly into the initial tree but, rather, adjoins first into

String Adjacency   Constituency   Predicative Adjunction   Modifier Adjunction
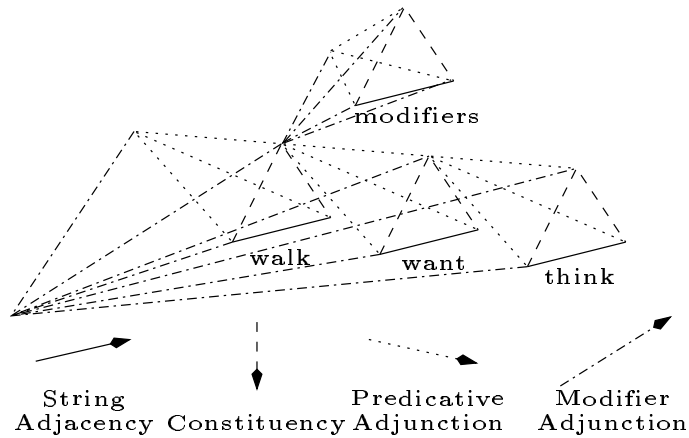
the AP with the resulting derived auxiliary tree adjoining into the initial tree. But the AP *yesterday* is compatible with both *walk* and *feed*. Thus the selectional restriction is no longer a local relationship but must stretch across the intervening tree. In this case, the derivation trees are too finely resolved. The order of adjunction is reflected in the structure of the derivation tree, but the selectional restrictions must hold between the modified VP and all of its modifiers regardless of the order in which they combine.

We can contrast this situation with that of *predicative adjunction*, the mechanism by which sentential complements are formed. Consider

    Cal thinks Bo wanted Asa to walk...

    Cal wants Bo to think Asa walked...

Here the initial tree is the tree for *walk* with auxiliary trees for *want* and *think* adjoining at its root. But in this circumstance the order of adjunction is critical. If we adjoin *think* into *want* we get the first order. If we adjoin them the other way we get the second. In both cases whether *walk* is tensed or not depends on the characteristics of the immediately local auxiliary tree—*think* takes a tensed complement, *want* takes an untensed complement.

So sometimes the order of adjunction is irrelevant and at other times it is critical. At least to this extent modifier and predicative adjunction are distinct processes. In [40] Schabes and Shieber proposed to accomodate this distinction by permitting multiple modifier trees to adjoin directly into the same node while limiting predicative auxiliary trees to a single adjunction at a given node. One of the difficulties of this approach is that the linear order of the modifiers becomes underspecified. While they must nest in some way in the derived tree, the order is not determined by the derivation tree.

We can achieve the same effect without losing the functional relationship between derivation and derived structures by interpreting modifier adjunction as concatenation in the fourth dimension. We allow modifier trees to combine, in the third dimension, only with other modifier trees. In doing so, they derive a compound modifier structure. This is then combined, in the fourth dimension, with the modified structure. Thus all modifiers are local, in the fourth dimension, to the modified word—immediate domination in the fourth dimension becomes the mediator of selectional restrictions on modifiers—but the structural relationships between the modifiers are fully resolved and potential semantic or pragmatic restrictions on their nesting can still be enforced via domination in the third dimension. Such an approach would give us four dimensional structures in which each dimension corresponds to a distinct type of syntactic relationship: string adjacency, constituency, predicative adunction and modification.

10 Conclusion

The results we have discussed are all based on a single hierarchy of multidimensional structures with uniform notions of language-theoretic and model-theoretic mechanisms. We have tried to show how, by varying the parameters of the structures—their dimension and branching factors—we can obtain useful characterizations of the classes of structures employed by a range of theories of syntax. One of the things that is remarkable, is the extent to which that range of theories covers "tree-based" theories of syntax. This is not, however, coincidental. These theories are "tree-based" because they share a common foundation in the notion that structural properties of language are direct reflections of some sort of constituency. The hierarchy of dimensions in our structures corresponds to a hierarchy of higher-order types of constituency. In some sense, the range of theories we treat can be seen as an exploration of the corresponding range of notions of constituency.

There are a variety of issues within this body of work which are yet to be resolved. From the purely theoretical perspective there is the question of whether the decidability results for the full second order theory of the one- and two-dimensional structures can be extended throughout the hierarchy of structures. While the details of this lift are more complicated (and yet unresolved) it seems clear that it will, in fact, go through. This would yield an infinite hierarchy of decidable theories that strictly extend SnS.

From the perspective of Computational Linguistics there are both theoretical and practical issues open. One question has to do with whether there are compelling linguistic motivations for moving to the higher order types of constituency. The third level of Weir's hierarchy (our fourth level) has shown up occasionally, for instance, in accounts of certain syntactic phenomena in

Hungarian or in mechanisms for generating paraphrases. But it is not clear yet whether these uses of the higher levels of the hierarchies are actually necessary or whether there is some theoretically plausible way of making sense of the higher-order constituency relationships they imply. From a practical perspective there is a question of whether the compilation of formulae to automata and the recognition process for the higher-dimensional automata can be feasibly implemented. Should this be the case, there is a very attractive possibility of using these mechanisms as a highly efficient and linguistically transparent method of developing and maintaining large-scale TAG (or higher-order) grammars.

References

[1] Tilman Becker. Patterns in metarules. In *Proceedings of the 3rd TAG+ Workshop*, Paris, 1994.

[2] Robert C. Berwick and Amy S. Weinberg. *The Grammatical Basis of Linguistic Performance*. MIT Press, 1984.

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

[4] Marie-Helene Candito. A principle-based hierarchical representation of LTAGs. In *Proceedings of COLING-96*, Copenhagen, 1996.

[5] N. Chomsky and M. P. Schüzenberger. The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, Studies in Logic and the Foundations of Mathematics, pages 118–161. North-Holland, Amsterdam, 2nd (1967) edition, 1963.

[6] Noam Chomsky. *Lectures on Government and Binding*. Foris Publications, Cinnaminson, NJ, 1981.

[7] Noam Chomsky. *Barriers*. MIT Press, 1986.

[8] Noam Chomsky. A minimalist program for linguistic theory. In *The View from Building 20*, pages 1–52. MIT Press, 1993.

[9] John Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4:406–451, 1970.

[10] Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–51, 1961.

[11] Roger Evans, Gerald Gazdar, and David Weir. Encoding lexicalized tree adjoining grammars with a nonmonotonic inheritance hierarchy. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguisitics (ACL'95)*, Cambridge, MA, 1995.

[12] Sandiway Fong. *Computational Properties of Principle-Based Grammatical Theories*. PhD thesis, MIT, 1991.

[13] Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar*. Harvard University Press, 1985.

[14] Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

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

[16] Saul Gorn. Explicit definitions and linguistic dominoes. In John F. Hart and Satoru Takasu, editors, *Systems and Computer Science, Proceedings of the Conference held at Univ. of Western Ontario, 1965*. Univ. of Toronto Press, 1967.

[17] Riny Huybregts. The weak inadequacy of context-free phrase structure grammars. In Ger J. de Haan, Mieke Trommelen, and Wim Zonneveld, editors, *Van Periferie Naar Kern*, pages 81–99. Foris Publications, Dordrecht, 1984.

[18] Mark Johnson. The use of knowledge of language. *Journal of Psycholinguistic Research*, 18(1):105–128, 1989.

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

[20] Aravind K. Joshi and Yves Schabes. Fixed and flexible phrase structure: Coordination in tree adjoining grammars. Presented at the DARPA Workshop on Spoken Language Systems, Feb. 1991. Asilomar, CA.

[21] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars and lexicalized grammars. In M. Nivat and A. Podelski, editors, *Tree Automata and Languages*, pages 409–431. Elsevier Science Publishers B.V., 1992.

[22] N. A. Khabbaz. A geometric hierarchy of languages. *jcss*, 8:142–157, 1974.

[23] D. Terrence Langendoen. On the weak generative capacity of infinite grammars. *CUN-YForum*, 1:13–24, 1976.

[24] Harry R. Lewis. *Unsolvable Classes of Quantificational Formulas*. Addison-Wesley, 1979.

[25] Maria Rita Manzini. *Locality: A Theory and Some of Its Empirical Consequences*. MIT Press, Cambridge, Ma, 1992.

[26] Michael A. Palis and Sunil M. Shende. Pumping lemmas for the control language hierarchy. *Mathematical Systems Theory*, 28:199–213, 1995.

[27] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, July 1969.

[28] Luigi Rizzi. *Relativized Minimality*. MIT Press, 1990.

[29] James Rogers. Weak MSO theories and control languages. In preperation.

[30] James Rogers. What does a grammar formalism say about a language. Technical Report IRCS-96-10, Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA, 1996.

[31] James Rogers. "Grammarless" Phrase Structure Grammar. *Linguistics and Philosophy*, 20:721–746, 1997.

[32] James Rogers. A unified notion of derived and derivation structures in TAG. In *Proceedings of the Fifth Meeting on Mathematics of Language MOL5 '97*, Saarbrücken, FRG, 1997.

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

[34] James Rogers. A descriptive characterization of tree-adjoining languages. In *Proc. of the 17th International Conference on Computational Linguistics (COLING'98) and the 36th Annual Meeting of the Association for Computational Linguistics (ACL'98)*, Montreal, 1998. ACL.

[35] James Rogers. On defining TALs with logical constraints. In Anne Abeillé, Tilman Becker, Owen Rambow, Giorgio Satta, and K. Vijay-Shanker, editors, *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 151–154, 1998.

[36] James Rogers. Generalized tree-adjoining grammar. In *Sixth Meeting on Mathematics of Language*, pages 189–202, 1999.

[37] James Rogers. wMSO theories as grammar formalisms. In *Algebraic Methods in Language Processing, Proceedings of the Second AMAST Workshop on Language Processing*, TWLT 16, pages 201–222, Iowa City, IA, 2000. Universiteit Twente.

[38] William C. Rounds. Mappings and grammars on trees. *Mathematical Systems Theory*, 4:257–287, 1970.

[39] Anoop Sarkar and Aravind Joshi. Coordination in TAG: Formalization and implementation. In *Proceedings of COLING'96*, Copenhagen, 1996.

[40] Yves Schabes and Stuart M. Shieber. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124, 1994.

[41] Stuart M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985.

[42] Martine Smets. Comparison of XTAG and LEXSYS grammars. In Anne Abeillé, Tilman Becker, Owen Rambow, Giorgio Satta, and K. Vijay-Shanker, editors, *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 159–163, 1998.

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

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

[45] K. Vijay-Shanker and Yves Schabes. Structure sharing in lexicalized tree-adjoining grammars. In *Proceedings COLING'92*, 1992.

[46] David J. Weir. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania, 1988.

[47] David J. Weir. A geometric hierarchy beyond context-free languages. *Theoretical Computer Science*, 104:235–261, 1992.

[48] Fei Xia, Martha Palmer, K. Vijay-Shanker, and Joseph Rosenzweig. Consistent grammar development using partial-tree descriptions for lexicalized tree-adjoining grammars. In Anne Abeillé, Tilman Becker, Owen Rambow, Giorgio Satta, and K. Vijay-Shanker, editors, *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 180–183, 1998.

[49] XTAG Research Group. A lexicalized tree adjoining grammar for English. Technical Report IRCS-98-18, Institute for Research in Cognitive Science, 1998.