Generalized Tree-Adjoining Grammar

James Rogers
School of Computer Science
Univ. of Central Florida
jrogers@cs.ucf.edu

Abstract

This paper continues a program extending results related to the descriptive characterization of the CFLs in terms of definability in the weak monadic second-order theory of trees to the TALs and the entire hierarchy of Weir's Control Languages. Previously, we have shown that the languages in this hierarchy can be characterized by definability in the weak monadic second-order theories of a generalization of trees to increasingly higher dimensions. Here, we explore the effect of admitting models with arbitrary finite branching. In the two-dimensional case we have shown previously that this admits tree sets recognized by infinite, but regular, tree-automata. As these sets correspond, in a very strong sense, to the sets of trees licensed by GPSG grammars we have referred to these automata as Generalized Finite-State Tree Automata and the sets they accept as Generalized Recognizable Sets of trees. In lifting this result to the third and higher dimension, we show that the definable sets of structures are exactly those recognized by d-dimensional tree automata which are, in essence, generalized recognizable sets of $(d-1)$-dimensional structures. In three-dimensions, these correspond to Tree-Adjoining Grammars in which the set of initial trees can be any generalized recognizable set of trees—a variation we refer to, by analogy, as Generalized Tree Adjoining Grammars. We show that this permits adoption of GPSG-style accounts of coordination which can potentially provide a conceptually clean account of coordinate constructions that have been problematic for TAG in the past.

1 Automata over Multi-Dimensional Trees

Over the last few years we have been exploring the generalization of long standing descriptive characterizations of the regular (Büchi, 1960; Elgot, 1961) and context-free (Doner, 1970; Thatcher and Wright, 1968) languages—characterizations in terms of model-theoretic definability over certain classes of structures—to the Tree-Adjoining Languages and beyond. The perspective that underlies this generalization is an analogy between TAGs as a tree-rewriting formalism and CFGs as a string rewriting formalism. Just as CFGs, in a standard conception, build trees by expanding nonterminal nodes into strings of child nodes—in essence attaching a depth-one tree at the frontier of a tree—TAGs can be conceived as building three-dimensional tree-like structures by expanding nodes (at which adjunction is permitted) into trees of child nodes (the adjoined auxiliary tree)—in essence attaching a depth-one pyramidal structure at the frontier of a similar structure of arbitrary depth. There is a direct and easy mapping between these three-dimensional derivation structures and the TAG derivation trees of the standard sort.[1] The two-dimensional yield of the set of these structures generated by a TAG is the set of trees the TAG generates; the one-dimensional yield of that set of trees is its string language.

At this point, it should be clear that this process iterates: as we increase the dimension of the structures we increase the complexity of the string languages. We can show that the hierarchy of languages generated in this way coincides

[1] Each node in the (two-dimensional) derivation tree is represented in the three-dimensional structure by the actual tree it names.

with Wier's Control Language Hierarchy (Weir, 1992). The analogy also extends in the other direction. If we consider a mechanism that expands nodes into single nodes—so that the rewrite rules are depth one single-dimensional structures—we get the regular languages.[2]

For all this to work smoothly we need a uniform hierarchy of domains on which to build these structures. Here, again, we start in the two-dimensional realm of trees and generalize in both directions. We take our trees to be labeled tree-domains in the sense of Gorn (1967). These are just sets of tree-addresses expressed as strings over the natural numbers. The address of the root of the tree is the empty string (which we will denote ε) and the addresses of the children of a node at address w are $w0, w1, \ldots$ in left-to-right order. A set of such strings is a well-formed tree-domain iff it is prefix closed (which corresponds to being downward closed wrt to domination) and if, whenever wi is in the set then wj for all $j < i$ is in the set as well (which corresponds to the addresses of each set of children being downward closed wrt less-than in their final element). In stepping down to one-dimensional structures we get string-domains which are simply sets of natural numbers that are downward closed wrt less-than (i.e., initial segments of $\mathbb{N}$ ordered by $<$).

The analogy is more uniform if we express the natural numbers in unary: the numeral for $i \in \mathbb{N}$ is a sequence of i '1's (1^i). Thus downward closure wrt to less-than becomes prefix closure wrt the sequences of '1's. A string-domain, then, is just a prefix closed set of sequences of '1's. A tree-domain is a set of sequences of sequences of '1's that is hereditarily prefix closed in the sense that the second-order sequences are prefix closed and, for any second-order sequence s in the tree-domain T, the set of first-order sequences $\{w \mid s \cdot \langle w \rangle \in T\}$ is also prefix closed.

To generalize this to higher dimensions, we note, first, that the address of a node in a tree-domain is exactly the sequence of string addresses one encounters in following the path in the tree from the root to that node. By analogy, then, the address of a node in a three-dimensional tree is the sequence of tree addresses one encounters in following the path in the three-dimensional structure from the root to that node. Consequently, such an address is a third-order sequence of '1's: a sequence of tree addresses which are themselves sequences of string addresses which are themselves sequences of '1's. A three-dimensional tree-domain is any set of third-order sequences of '1's which is hereditarily prefixed closed and, in general, a d-dimensional tree-domain is any d^{th} -order sequence of '1's that is hereditarily prefix closed. We will denote the class of all d^{th} -order sequences of '1's as ${}^d1$ and the class of all d-dimensional tree-domains as $\mathbb{T}^d$. Thus $\mathbb{T}^1 \subseteq {}^11 = 1^*$ is the class of string domains, $\mathbb{T}^2 \subseteq {}^21 = (1^*)^*$ is the class of ordinary tree-domains, etc. We will refer to individual structures in the class $\mathbb{T}^d$ as 'Td's—ordinary trees-domains are 'T2's and string-domains are 'T1's.[3] We will admit empty Td, which we denote '$\emptyset$'.

It is critical to be as clear as possible when discussing these higher-order sequences. We will generally employ w, v, etc. to denote sequences of a given order and s, t, etc. for sequences of the next higher order. Concatenation ('$\cdot$') will always be an operation on sequences of the same order. Thus $s \cdot \langle w \rangle$ denotes the $(d+1)$-order sequence in which the final d-order sequence is w. (To be consistent, when we referred to the sequence wi above, we should, perhaps, have referred to $w \cdot \langle i \rangle$.) We will also employ p to denote sequences of the next higher order than s when necessary. The residual ambiguity arises from the fact that the zero-length sequence of any order is just 'ε'. Thus, 'ε' denotes the zero length sequence of any order, '$\langle \varepsilon \rangle$' denotes the length one sequence of any order greater than one (containing a single zero-length sequence of the next lower order), etc. The orders of sequences within a sequence are always uniform. Thus, while '$\langle \varepsilon, \varepsilon \rangle$' denotes a length

[2] If we take the rules to be zero-dimensional structures they license no rewriting at all and we get exactly the finite languages.

[3] We have referred, in the past, to these structures as *Tree Manifolds*, but as a hierarchy of equivalent structures (with considerably different presentation) has been previously introduced under the term multi-dimensional trees (Baldwin and Strawn, 1991) (unfortunately this is far from the only class of structures that have been termed such) we will adopt that terminology, using 'multi-dimensional tree-domain' when we wish to be clear about the specific structures on which our notion is based.

two sequence in which both 'ε' denote empty sequences of the same order, '$\langle\varepsilon,\langle\varepsilon\rangle\rangle$' denotes a length two sequence containing a zero length sequence of the next lower order and a length one sequence of that same order itself containing a zero length sequence of the next lower order yet—the two occurrences of 'ε' denote sequences of different orders. The actual order intended will always either be clear from the context or will be intentionally unspecified—thus the root of a structure of any order is always at address 'ε'.

A Σ-*labeled Td* is a pair $\langle T,\tau\rangle$ where T is a d-dimensional tree-domain and $\tau : T \to \Sigma$ is an assignment of labels in Σ to nodes in T. We will denote the class of all Σ-labeled Td as $\mathbb{T}^d_\Sigma$. We will use '$\emptyset$' to denote empty Σ-labeled Td as well.

1.1 Td Grammars and Automata

A *Td Grammar* is just a finite set of local (depth one in their major dimension) Σ-labeled Td, licensing the expansion of nodes labeled identically to the root into a Σ-labeled T$(d-1)$ of children. We will take these to be presented as a set of pairs consisting of the label of the root and the child structure:

$$G^d \subseteq \Sigma \times \mathbb{T}^{d-1}_\Sigma, \quad \text{finite}\ .$$

A Σ-labeled Td licensed by such a grammar, relative to some distinguished set of initial symbols, iff its root is labeled with one of the initial symbols and every local Td it includes is contained in the grammar:

$$\begin{aligned}&\mathcal{G}(\Sigma_0) \stackrel{\text{def}}{=}\\ &\quad\{\mathcal{T} = \langle T,\tau\rangle \mid T \text{ finite}, \tau(\varepsilon)\in\Sigma_0,\\ &\qquad\text{and } (\forall s\in T)[\langle\tau(s),\langle T,\tau\rangle\,|_{\mathrm{Ch}(T,s)}\rangle \in \mathcal{G}]\}.\end{aligned}$$

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

$$\begin{aligned}&\langle T,\tau\rangle\,|_{\mathrm{Ch}(T,s)} \stackrel{\text{def}}{=}\\ &\quad\langle \mathrm{Ch}(T,s), \{w \mapsto \tau(s\cdot\langle w\rangle) \mid w\in\mathrm{Ch}(T,s)\}\rangle\ .\end{aligned}$$

In the two-dimensional case this is, in essence, a presentation of the the set of derivation trees of a CFG as a *local set* of trees (Gécseg and Steinby, 1984) and we will adopt this terminology for all levels.

Definition 1 *A set of Σ-labeled Td is* local *iff it is $\mathcal{G}^d(\Sigma_0)$ for some Td grammar $\mathcal{G}^d$ and set of initial labels Σ_0.*

This generalizes the CFGs in two ways. First, we allow for multiple start symbols, a common relaxation, particularly when considering sets of derivation trees. Second, we do not distinguish terminals and non-terminals in the normal sense: any node can be expanded if it is labeled with a symbol labeling the root of local Td in $\mathcal{G}^d$ in which the child structure is non-empty. On the other hand, every local Td in the licensed structure must be explicitly licensed by $\mathcal{G}^d$ and this includes the nodes on the frontier for which the child structures are empty. A symbol $\sigma\in\Sigma$ may label a node on the frontier of the structure iff it is licensed by a local tree in $\mathcal{G}^d$ with an empty child structure: $\langle\sigma,\emptyset\rangle$. In this way the set of terminal symbols is distinguished but, since σ may license both empty and non-empty child structures they may, potentially, be rewritten. Of course, neither of these generalizations affects the class of string languages these trees yield.[4]

Just as we use the two-dimensional case, CFGs, as the model for our grammars, we use finite-state tree-automata as the model for our automata. These can be understood as a mechanism that licenses Σ-labeled trees on the basis of an assignment of states, drawn from a finite set, to the nodes in the tree. The states assigned to the frontier nodes and the root are distinguished and the states assigned to each local tree as well as the label assigned to its root are constrained. Alternatively, they can be understood as a CFG generating trees labeled with a state set Q along with a mapping from Q to Σ. A *Td Automaton* with state set Q and label set Σ is a finite set of triples:

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

[4] We can restrict these grammars to the traditional CFGs by requiring Σ_0 to be singleton and by requiring the set of symbols licensed to label nodes with no children to be disjoint from those licensed to label nodes with children. Conversely, we can convert any of these grammars into CFGs of the traditional sort that generate the same string language by adding a new start symbol which rewrites only to the symbols in Σ_0 and by distinguishing two variants of each symbol, one that may (potentially) label interior nodes and one that may (potentially) label leaves.

The interpretation of a tuple $\langle \sigma, q, \mathcal{T} \rangle \in \mathcal{A}^d$ is that if a node of a Td is labeled σ and $\mathcal{T}$ encodes the assignment of states to its children, then that node may be assigned state q.[5] A *run* of an Td automaton $\mathcal{A}$ on a Σ-labeled Td $\mathcal{T} = \langle T, \tau \rangle$ is an assignment $r : T \to Q$ of states in Q to nodes in T in which each assignment is licensed by $\mathcal{A}$. Note that this implies that a maximal node (wrt to the major dimension) labeled σ may be assigned state q only if there is a tuple $\langle \sigma, q, \emptyset \rangle \in \mathcal{A}^d$. If we let $Q_0 \subseteq Q$ be any set of *accepting states*, then the set of (finite) Σ-labeled Td recognized by $\mathcal{A}$, relative to Q_0, is that set for which there is a run of $\mathcal{A}$ that assigns the root a state in Q_0:

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

Definition 2 *A set of Σ-labeled Td is* recognizable *iff it is $\mathcal{A}(Q_0)$ for some Td automaton $\mathcal{A}$ and set of accepting states Q_0.*

The classes of recognizable sets strictly include the corresponding classes of local sets. The set of $\{a, b\}$-labeled n-branching structures in which exactly one node is labeled 'a' is, for instance, recognizable but not local. The relationship between the mechanisms is, perhaps, clearer if one views the automata as a variety of grammar that licenses two structures: a Q-labeled Td (which determines the structure) along with an isomorphic Σ-labeled Td (the actual recognized structure). It is important to note that the mapping between these structures associates labels in Σ with local Q-labeled Td, not with the label of the root alone. We can turn this into a genuine homomorphism if we take the basic structure to be a $\Sigma \times Q$-labeled tree. In which case the homomorphism is simply a projection. In this way, the characterizations of the regular sets of strings and recognizable sets of trees in terms of the corresponding local sets (originally due to Chomsky and Schützenberger (1963) and Thatcher (1967)) generalize easily.

Lemma 1 *A set of Σ-labeled Td is recognizable iff it is the projection of a local set.*

Just as the two-dimensional grammars are equivalent to CFGs, the two-dimensional automata are equivalent to the finite-state tree-automata. In the one-dimensional case we get ordinary (non-deterministic) finite-state automata over strings. In the three-dimensional case we get a slight generalization of Tree-Adjoining Grammars with adjoining constraints.

Definition 3 *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.*

Adjoining, in such a TAG, is completely controlled by the adjoining constraints.[6] The sets recognized by T3 automata are exactly the sets of labeled T3 that correspond to the sets of derivation trees of non-strict TAGs with adjoining constraints. (Rogers, 1997)

For the notion of yield to be well-defined at dimensions three and higher we need a mechanism that determines the way in which the maximal structures of the next lower dimension splice together. For T3 grammars and automata the issue that needs to be determined is, in TAG terminology, which node is the "foot" of the tree. To settle this, we will assume classes of distinguished states, one for each dimension, that pick out exactly one order d head node in each Td child structure—one order 1 head in each string of children in every local tree, one order 2 head in each tree of children in every local T3, etc. The node corresponding to the foot of a

[5] This is a "bottom-up" interpretation. There is an analogous "top-down" interpretation, but for all $d \geq 2$, Td automata that are deterministic under the top-down interpretation are strictly weaker than those that are non-deterministic, while those that are deterministic under the bottom-up interpretation are equivalent to the non-deterministic variety. It should be emphasized that the only place the distinction between top-down and bottom-up arises is in the definition of determinism. These automata are interpreted purely declaratively, as licensing assignments of states to nodes.

[6] Such a relaxation has been employed in a number of contexts. An example is Feature-structure based TAG (FTAG), the variant most commonly employed in applications, where it is not entirely clear what distinguishes the labels which select auxiliary trees from the other features of the nodes which are only required to agree up to unifiability.

structure is the maximal node (in the d^{th} dimension) one reaches by following the path of $(d-1)$ order heads from the root. Using this mechanism to define the (2-dimensional) yield, we get that the set of trees yielded by the recognizable sets of T3 are exactly the sets of trees generated by non-strict TAGs with adjoining constraints. Moreover, we can show that the hierarchy of classes of string languages yielded by the recognizable sets of Td coincide with the classes of Weir's hierarchy of *Control Languages* (1992).[7]

2 wSnTd, $n < \omega$

The key strength of this approach is that, as with the characterization of the relationship between the local and recognizable sets, essentially every result for regular languages lifts uniformly to all levels of the hierarchy. The dimension of the objects being manipulated becomes a parameter of the proof playing no essential role in the reasoning it employs. In particular we get that the classes are closed under determinization (given a suitable notion of *deterministic*), Boolean operations, projection and cylindrification ("inverse" projection) and that emptiness of the recognizable sets is decidable.

These results allow us to lift the characterizations of the regular languages in terms of the weak monadic second-order theory of the natural numbers ordered by less-than (wS1S) (Büchi, 1960; Elgot, 1961) and of the recognizable sets of trees in terms of the weak monadic second-order theory of the complete n-branching tree (wSnS) (Doner, 1970; Thatcher and Wright, 1968). Let T_n^d be the *complete n-branching Td*—that in which every point has a child structure that has depth n in all its dimensions other than d. Let

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

where, for all $x, y \in T_n^d$:

$$\begin{aligned}
x \lhd_d y &\iff y = x \cdot \langle p \rangle, \quad p \in {}^{d-1}1 \\
&\vdots \\
x \lhd_2 y &\iff x = p \cdot \langle \cdots \langle s \rangle \cdots \rangle \\
&\quad \text{and } y = p \cdot \langle \cdots \langle s \cdot \langle w \rangle \rangle \cdots \rangle, \quad w \in 1^* \\
x \lhd_1 y &\iff x = p\langle \cdots \cdot \langle s \cdot \langle w \rangle \rangle \cdots \rangle \\
&\quad \text{and } y = p \cdot \langle \cdots \langle s \cdot \langle w \cdot 1 \rangle \rangle \cdots \rangle, \quad w \in 1^*,
\end{aligned}$$

which is to say that $x \lhd_i y$ iff x is the immediate predecessor, or *parent*, of y in the i^{th}-dimension.

The *weak monadic second-order language* of T_n^3 includes, in addition to symbols for the parent relations and the usual logical connectives and quantifiers, two sorts of variables: those ranging over individuals (x_1, y_1, etc.) and those ranging over *finite* sets (X_1, Y_1, etc.). We will say

$$\mathsf{T}_n^d \models \varphi(x_1, \ldots, x_n, X_1, \ldots, X_m)\ [\boldsymbol{s}]$$

iff φ is a formula with free variables among the x_i and X_i and $\boldsymbol{s}$ is an assignment of individuals and *finite* sets of individuals to those variables which makes φ true in T_n^d. The set of all sentences of this language that are satisfied by T_n^d is the *weak monadic second-order theory of* T_n^d, denoted wSnTd.[8]

Definition 4 *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 of a Td) and X_σ for each $\sigma \in \Sigma$ (interpreted as the set of σ-labeled points in T), such that*

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

It should be reasonably easy to see how any recognizable set can be defined in this way. The converse can be obtained by a simple lift of the corresponding proofs for S1S and SnS. (Rogers, 1998a)

Theorem 1 *A set of labeled Td is definable in wSnTd, $n < \omega$, iff it is recognizable.*

2.1 "Non-local" Relations

The strictly local nature of the immediate parent relations makes them inconvenient for expressing many syntactic relationships. Part of the strength of the characterization of recognizable sets in terms of the monadic second-order theories is that, as transitive closure is definable in these theories, we can move to the non-local versions of these relations without increasing the complexity of the sets we define.

[7] Weir's initial class is the CFLs which correspond to our third level T2.

[8] wS1T1 is equivalent to wS1S in the sense of interinterpretability, as is wS1Td for all d. wSnT2 is interinterpretable with wSnS for all $n \ge 2$.

$$
\begin{aligned}
h(\varepsilon) &= \varepsilon \\
h(\langle\varepsilon\rangle) &= h(\varepsilon)\cdot\langle 1\rangle^0\cdot\langle\varepsilon\rangle \\
&= \varepsilon\cdot\varepsilon\cdot\langle\varepsilon\rangle = \langle\varepsilon\rangle \\
h(\langle 11\rangle) &= \varepsilon\cdot\langle 1\rangle\cdot\langle 1\rangle\cdot\langle\varepsilon\rangle \\
&= \langle 1,1,\varepsilon\rangle \\
h(\langle 1,11\rangle) &= \langle 1,\varepsilon\rangle\cdot\langle 1\rangle\cdot\langle 1\rangle\cdot\langle\varepsilon\rangle \\
&= \langle 1,\varepsilon,1,1,\varepsilon\rangle
\end{aligned}
$$

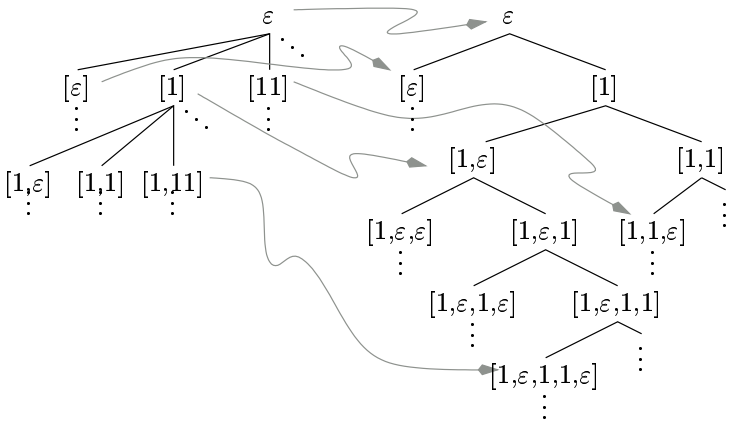

Figure 1: Embedding T^2_ω in T^2_2

Definition 5 *A* Σ-Labeled Headed Td *is a structure:*

$$\langle T_n, \lhd_i^+, H, P_\sigma\rangle_{1\le i\le d,\,\sigma\in\Sigma},$$

where T_n *is a rooted, connected subset of* T_n^d *for some* n*, and* $\lhd_i^+$ *is the irreflexive transitive closure of* $\lhd_i$*, inherited in the sense that if* $x \lhd_i^+ y$ *and* $x \lhd_{i+1}^+ z$ *and every point on the path between* x *and* z *in the* $i+1^{\text{st}}$ *dimension lies on the* spine *of its* i^{th} *dimensional structure (the sequence of* $i-1$ *order heads from its root to its yield) then* $z \lhd_i^+ y$*.*

For a formal definition of this relation see (Rogers, 1998a). Intuitively, in the two-dimensional case it extends linear precedence over dominance in the usual way, in the three-dimensional case it, in essence, says that a node in an adjoined tree dominates those nodes dominated by the node at which it is adjoined iff it dominates the foot of the adjoined tree.[9]

Theorem 2 *A set of labeled Td is recognizable iff it is weak monadic second-order definable as a set of labeled headed Td.*

3 wSωTd

Since automata and grammars are restricted to be finite the branching factor of the sets they license is finitely bounded. But all that bounds the branching factor of the sets of models definable in the signature of wSnTd is the fact that we interpret them as subsets of a structure that has finitely bounded branching. In the two-dimensional case, linguistic motivation for admitting sets of models with unbounded (but still finite) branching is well established, particularly in "flat" accounts of coordination. On the model-theoretic side, we can accommodate such accounts by moving to finite subsets of the ω-branching structure T^d_ω, that is, to wSωTd. By lifting a proof of Rabin's (Rabin, 1969) we can show that this does not increase the descriptive power.

Lemma 2 (from Rabin) *There is an effective translation* $\varphi \mapsto \varphi'$ *such that, for all* $n \le \omega$*,* $\varphi \in$ wSnTd *iff* $\varphi' \in$ wS2Td*.*

The translation is based on an embedding h of T^d_ω into T^d_2. This is easiest to visualize in the two dimensional case, where it is a standard embedding of arbitrarily branching trees into binary branching trees (Figure 1): the image of the k^{th} child of a node is found by starting at the image of that node and following the path that takes the right child k times and then the left child once. In terms of T2, this says that $h(s\cdot\langle 1^k\rangle)$ is $h(s)\cdot\langle 1\rangle^k\cdot\langle\varepsilon\rangle$. In essence, we raise the type of the sequence of '1's to an equal length sequence of '$\langle 1\rangle$'s and add '$\langle\varepsilon\rangle$'.

We generalize this in two steps. First, assume that the structure is two branching in all but its

[9] One can be flexible about which relations to include in the signature of these structures. In (Rogers, 1998a) we explicitly include both $\lhd_i$ and $\bar{\lhd}_i$—the extension of $\lhd_i$ to local structures. As each of these is definable, in the monadic second-order language, from the others the choice can be dictated by convenience.

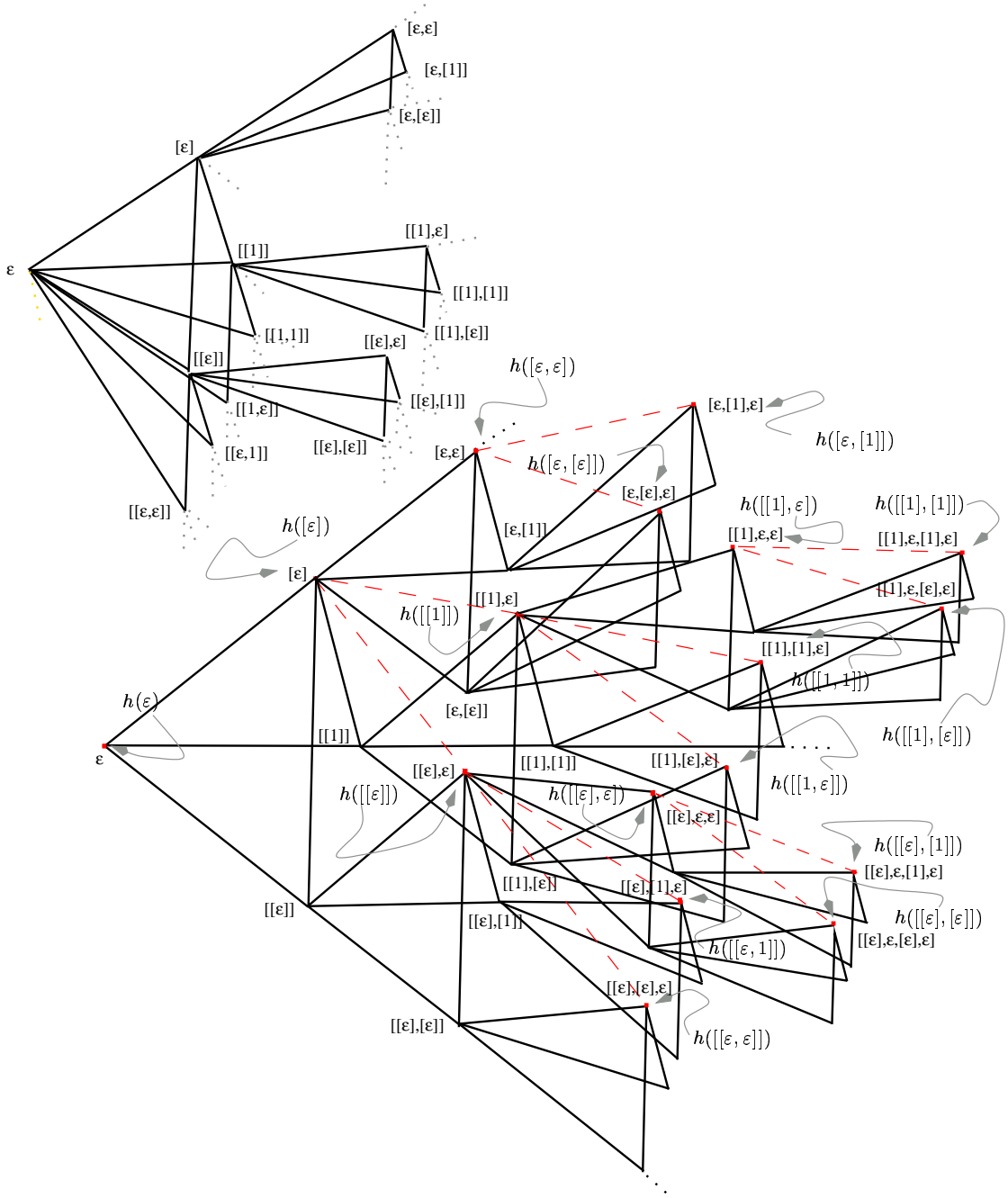

Figure 2: Embedding the fragment of T^3_ω which is two-branching in the first dimension into T^3_2

highest two dimensions. Let:

$$\begin{aligned}
\mathrm{Raise}(\varepsilon) &\stackrel{\text{def}}{=} \varepsilon \\
\mathrm{Raise}(w \cdot 1) &\stackrel{\text{def}}{=} \mathrm{Raise}(w) \cdot \langle 1 \rangle \\
\mathrm{Raise}(s \cdot \langle w \rangle) &\stackrel{\text{def}}{=} \mathrm{Raise}(s) \cdot \langle \langle w \rangle \rangle \\
h(w) &\stackrel{\text{def}}{=} w, \qquad w \in 1^* \\
h(s \cdot \langle w \rangle) &\stackrel{\text{def}}{=} h(s) \cdot \mathrm{Raise}(w) \cdot \langle \varepsilon \rangle
\end{aligned}$$

The function Raise raises the type of a sequence s; h leaves simple sequences unchanged and, for higher-order sequences, first raises the type and then adds the $\langle \varepsilon \rangle$. An example of the embedding in the three-dimensional case is illustrated in Figure 2. This map has two key properties. First, the image of this two-branching

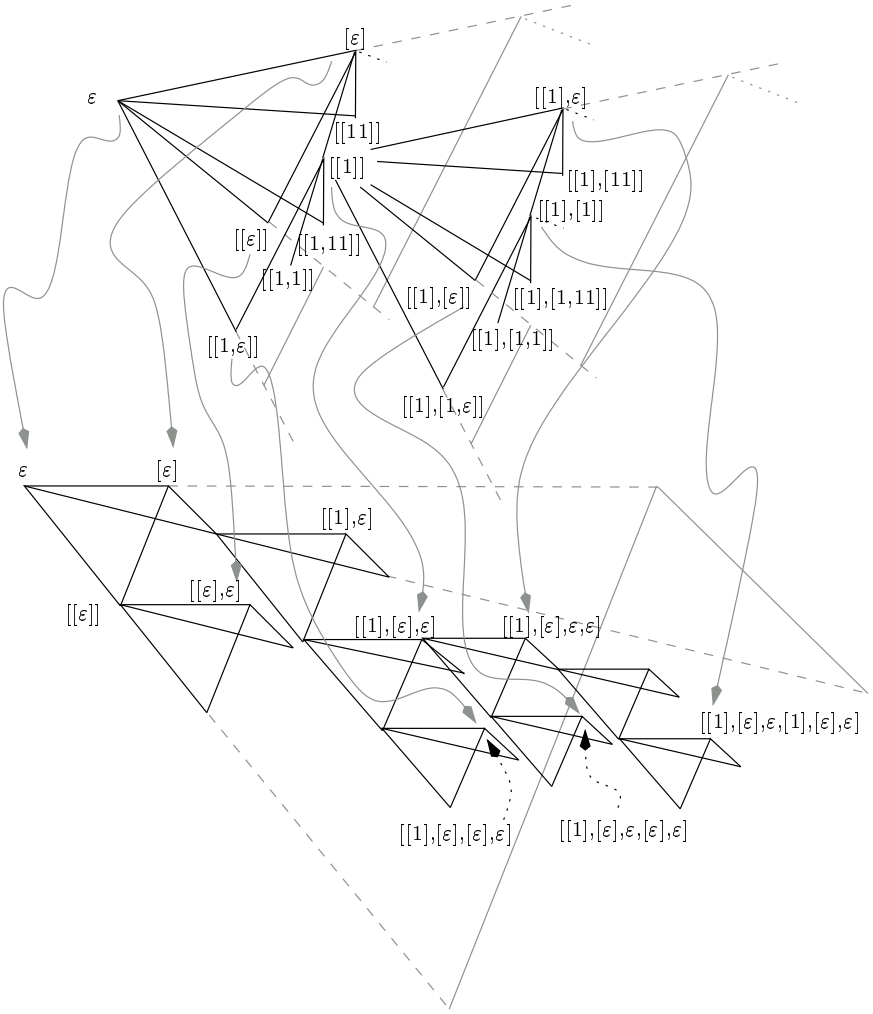

Figure 3: Embedding T^3_ω in T^3_2

subset of T^d_ω in T^d_2 is exactly the set containing its (d-dimensional) root and the (($d-1$)-dimensional) roots of its $(d-1)$-dimensional child structures, i.e. the set of all points that are minimal wrt $\lhd_{d-1}$—clearly a definable subset. Secondly, the map preserves $\lhd_i^+$. Thus every property of subsets of T^d_ω definable in $\langle T_\omega, \lhd_i^+, H, P_\sigma\rangle_{1\le i\le d,\,\sigma\in\Sigma}$, which is to say every property definable in wSωTd, is definable in the restriction of $\langle T_2, \lhd_i^+, H, P_\sigma\rangle_{1\le i\le d,\,\sigma\in\Sigma}$ to points which have no sibling in the $(d-1)$-dimension that properly dominates them. The lemma then follows by a simple induction on d.

The inductive step can be incorporated into the map simply by recursively embedding each element of higher-order sequences in their corresponding two-branching structures before raising their type:

$$\begin{aligned}\hat{h}(w) &\stackrel{\text{def}}{=} w \qquad w \in 1^*\\ \hat{h}(s\cdot\langle w\rangle) &\stackrel{\text{def}}{=} \hat{h}(s)\cdot \text{Raise}(\hat{h}(w))\cdot\langle\varepsilon\rangle\end{aligned}$$

The three-dimensional case is illustrated in Figure 3

The range of $\hat{h}$ is a definable subset of the range of h and is, therefore both definable and preserves $\lhd_i^+$.

3.1 Generalized Recognizable Sets

In (Rogers, 1998b) we show that, for each $\sigma \in \Sigma$, the set of strings labeling the children of nodes labeled σ in a set of trees definable in wSωT2 is a regular set. The proof involves extracting a grammar for the set (of strings) from

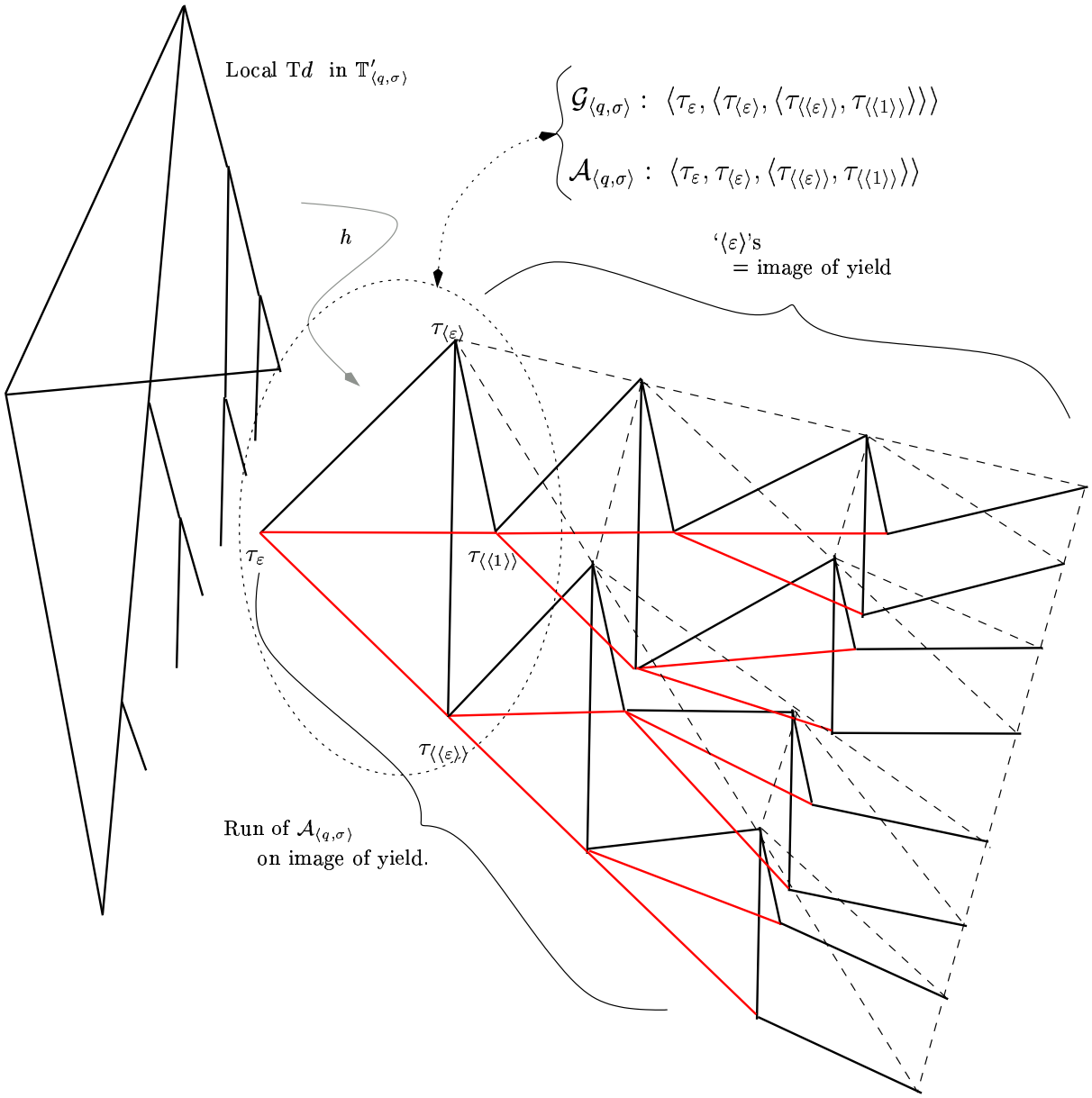

Figure 4: Recognizability of T2 yields of recognizable sets of T3.

the automaton accepting the image of the set trees in T_2^2 which turns out to be right-regular. It follows, then, that every such set of trees is accepted by a T2 automaton that is infinite but regular. In general, we refer to the class of all finitely presentable Td automata as the class of *Generalized Td Automata*. In practice, we will restrict our attention to generalized Td automata in which, for each label/state pair, the set of T$(d-1)$ expanding nodes with that label and state is a *Generalized Recognizable Set*, that is, a set that is itself recognized by a generalized T$(d-1)$ automaton.[10]

The corresponding grammars, in the two-dimensional case, have been referred to as *extended* CFGs by Thatcher (Thatcher, 1967) and as *hypergrammars* by Langendoen (Langen-

[10] At the $d = 1$ level one has that the local, recognizable, and generalized recognizable sets of the next lower order are all just the finite sets. Thus the recognizable and generalized recognizable sets of strings coincide with the regular sets.

doen, 1976). As we note in (Rogers, 1998b), there is a strong parallel between the sets recognized by generalized tree-automata and the sets of trees licensed by GPSG grammars: in the regularity of the branching, in the capacity to distinguish similarly labeled nodes via a finite set of features, and in the ability to separate constituency constraints ($\lhd_2$) from precedence constraints ($\lhd_1$). Our preferred terminology emphasizes this connection.

That every generalized recognizable set of Td is definable in wSωTd can be established by induction on d. To lift the other direction of the characterization, suppose that $\mathbb{T}$ is a set of Σ-labeled Td definable in wSωTd. Again assume, for induction, that the Td in $\mathbb{T}$ are 2-branching in all but the d^{th} and $(d-1)^{\text{st}}$ dimensions. By the lifted version of Thatcher's Lemma, $\mathbb{T}$ is a projection of a local set of Td. Let $\mathbb{T}'$ be the local set constructed in the proof of the lemma. Let $\mathbb{T}'_{\langle q,\sigma\rangle}$ be the set of local Td occurring in $\mathbb{T}'$ which are rooted at nodes labeled $\langle q,\sigma\rangle$. Clearly, this set is wSωTd definable; we need to establish that the set of T$(d-1)$ it yields is a generalized recognizable set. By the lemma we have that $h(\mathbb{T}'_{\langle q,\sigma\rangle})$ is definable in wS2Td and is, thus, a recognizable set. Moreover, because h preserves $\lhd_i^+$, the $(d-1)$-dimensional yield of $h(\mathbb{T}'_{\langle q,\sigma\rangle})$ is the same as that of $\mathbb{T}'_{\langle q,\sigma\rangle}$. By the lifted version of Thatcher's lemma, again, this is also the yield of $\mathbb{T}''$, a local set of 2-branching Td. Let $\mathcal{G}_{\langle q,\sigma\rangle} \subseteq \Sigma' \times \mathbb{T}^{(d-1)}_{\Sigma'}$ be the Td grammar licensing that set. The pairs in $\mathcal{G}_{\langle q,\sigma\rangle}$ represent Σ'-labeled local Td: Td that are 2-branching in all dimensions. These consist of a root ε and a child T$(d-1)$ which, itself, consists of a root $\langle\varepsilon\rangle$ and a child T$(d-2)$, with the $(d-1)$ dimensional yield of $h(\mathbb{T}'_{\langle q,\sigma\rangle})$ being made up of the $\langle\varepsilon\rangle$ points exclusively. (For an illustration of the 3-dimensional case see Figure 4.) The idea is to convert $\mathcal{G}_{\langle q,\sigma\rangle}$ into a T$(d-1)$ automaton (in which Σ' serves both as the set of labels and the set of states) that recognizes the $(d-1)$ dimensional yield of $h(\mathbb{T}'_{\langle q,\sigma\rangle})$. Let

$$\mathcal{A}_{\langle q,\sigma\rangle} \stackrel{\text{def}}{=} \{\langle \tau(\varepsilon), \sigma', \langle T,\tau\rangle \,|_{\mathrm{Ch}(T,\varepsilon)}\rangle \mid \langle \sigma', \langle T,\tau\rangle\rangle \in \mathcal{G}_{\langle q,\sigma\rangle}\}.$$

This, in effect, interprets the pairs from $\mathcal{G}_{\langle q,\sigma\rangle}$, consisting of a label σ' and a Σ'-labeled T$(d-1)$ $\langle T,\tau\rangle$ as a triple consisting of a label σ', a state $\tau(\varepsilon)$ and the Σ'-labeled T$(d-2)$ yield of $\langle T,\tau\rangle$. Thus, it is a T$(d-1)$ automaton. To see that it recognizes the yield of $h(\mathbb{T}'_{\langle q,\sigma\rangle})$ one need only to note that the Σ'-labeled T$(d-1)$ made up of the non-'$\langle\varepsilon\rangle$' nodes of a T$d$ in $\mathbb{T}''$ and the Σ'-labeled T$(d-1)$ made up of the '$\langle\varepsilon\rangle$' nodes (that is, the image under h of the yield of the tree in $\mathbb{T}'_{\langle q,\sigma\rangle}$) are parallel T$(d-1)$ with the former being an accepting run of $\mathcal{A}_{\langle q,\sigma\rangle}$ on the latter.

3.2 Generalized Tree-Adjoining Grammar

Just as definability in wSωT2 captures the sets of trees admitted by GPSG style grammars—CFGs in which the rhs of productions may be regular sets—definability in wSωT3 captures a generalization of (non-strict) TAG in which the set of initial trees may be any generalized recognizable set—the direct analog of GPSG in the domain of trees.[11] Moreover, working within wSωT3 allows factorization of independent constraints as in the separation of dominance and precedence constraints in GPSG. Consequently, we will refer to axiomatic definitions of syntactic structures in the language of wSωT3 as Generalized Tree-Adjoining Grammars.

4 Coordination in Generalized TAG

Given that one of the attributes of GPSG is its elegant account of coordination, this is the obvious place to look for potential usefulness of the added power of Generalized TAG. While the choice of structures must, in practice, be motivated on linguistic grounds—which we make no attempt to address—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.

4.1 Ordinary Conjunction

The XTAG grammar (Group, 1998) treats coordination in two cases. Non-VP coordination is accomplished with a family of auxiliary trees anchored by conj adjoining [conj X] to the right of X (for X in Adj, A, P, PP, N, NP, Det,

[11] A similar sort of extension, known as *Schema-TAGs*, shows up in (Harbusch et al., 1998). These were evidently introduced by Weir in his dissertation proposal although they do not appear in the dissertation itself. (Weir, 1988)

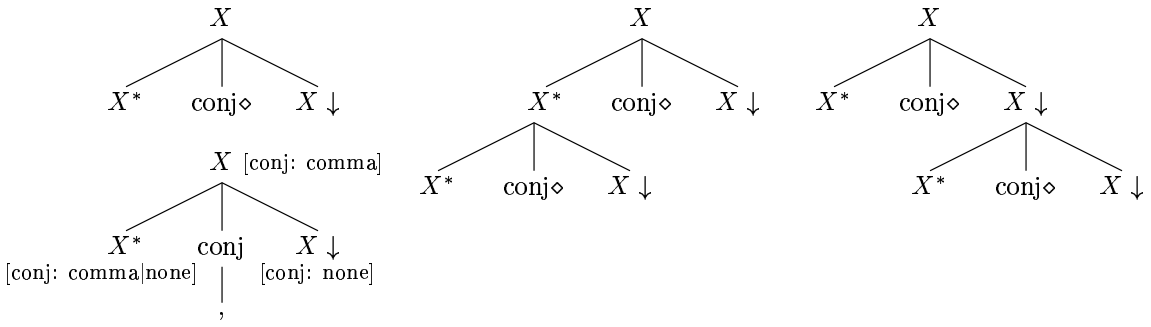

Figure 5: Non-verb coordination in XTAG

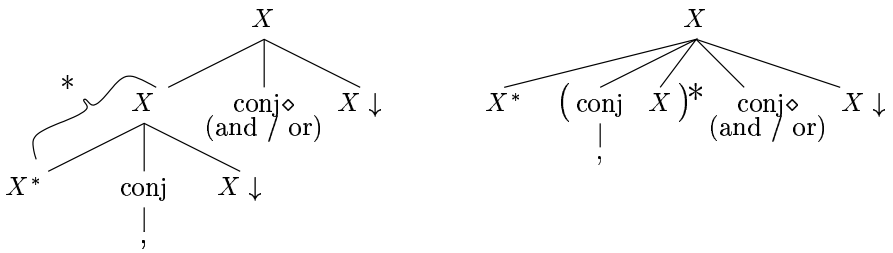

Figure 6: Non-verb coordination in Generalized TAG

or S) (Figure 5) 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 overgeneration 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 6). 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, yielding a range of syntactic phenomena from a small set of interacting constraints. Thus one can, if one is so inclined, import the GPSG account of the syntax of coordination directly into Generalized TAG.

From a theoretical perspective, one objection to adopting such an account is that these iterated structures no longer appear to capture minimal recursive constructions. However, it seems more plausible to analyze a construction like "neither Alan nor Barbara nor Carl" as a single conjunction than as a nested structure. This suggests that it might be reasonable to base the notion of what constitutes a minimal construction at least in part on its semantics rather than purely on its structure.

From the practical point of view the issue is more likely to be concern over the effect that such an approach may have on the difficulty of parsing. Here, though, at least in the case of iterated constructions like these, it seems unlikely that there would be any effect on the asymptotic complexity and very likely little effect on actual performance: adding iterated initial trees simply adds an alternative between closing a tree or projecting it to another iteration (or between

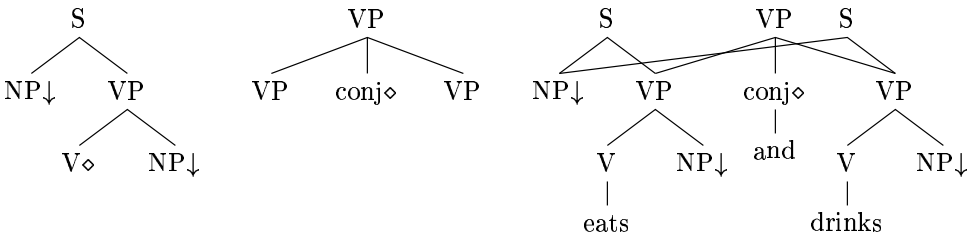

Figure 7: Predicative coordination in XTAG

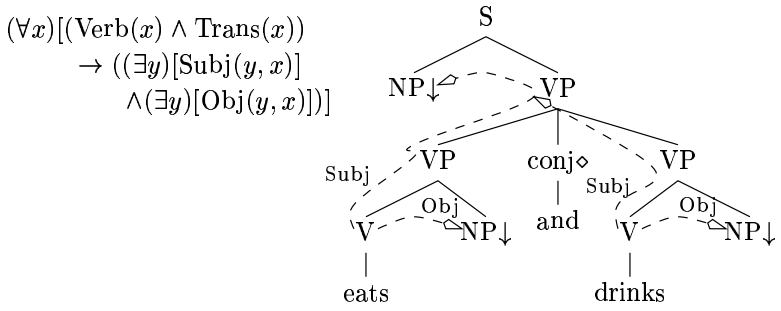

Figure 8: Predicative coordination in Generalized TAG

projecting in one way or another).

4.2 Predicative Coordination

VP coordination, such as gapping and right-node-raising constructions, have always been problematic in TAG in that, while TAG requires 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 (Joshi and Schabes, 1991) proposed to handle this with a mechanism that merges trees, collapsing their common structure. (Figure 7.) Sarkar and Joshi (Sarkar and Joshi, 1996) 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 derivation tree itself is of more interest than the structure it describes; the fact that it 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 (Rogers, 1998c) this allows modularization of the grammar of the sort suggested by Vijay-Shanker and Schabes (Vijay-Shanker and Schabes, 1992) and developed variously by Becker (Becker, 1994), Evans and Weir and others at Sussex (Evans et al., 1995; Smets, 1998), Candito (Candito, 1996), and Xia, Palmer, Vijay-Shanker and Rosenweig. (Xia et al., 1998) 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 8.) The set of elementary trees selected by a lexical item in the traditional presentation of a LTAG grammar is the set of all trees satisfying the constraints it selects in this presentation.[12] Under these circumstances, the requirement that the linguistically significant relationships between a predicate and its arguments be expressed in a single elementary structure (here a set of constraints) is not inconsistent with the possibility

[12] In practice one is likely to pre-compile these.

that, under some circumstances, in coordinate constructions for instance, their syntactic realization might be shared between multiple predicates.

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 wSωT3 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.

5 Conclusions

We have argued, previously, that by employing wSnT3 as a sort of higher-level language, one can define TAGs directly in terms of the linguistic relationships they are meant to express and that, in doing so, one gets an extreme degree of modularization of the grammar, abstraction away from the mechanical details of issues like feature passing, and a clear expression of the linguistic theory the TAG embodies. In this paper we have explored the consequences of relaxing the finite bound on n—an artificial restriction from the model-theoretic point of view. We have shown that the classes of sets of structures definable in the general theories wSωTd correspond to those recognized by a natural hierarchy of infinite, but finitely presentable, automata and that they correspond, in a strong sense, to the sets of structures licensed by GPSG-style grammars raised to arbitrary dimension. We refer, then, to grammars defined in wSωT3 as Generalized TAGs. In exploring the utility of the generalization, we have taken an initial look at the syntax of coordination and show not only that a GPSG-style account of coordination can be imported into Generalized TAG essentially intact, but that this has the potential to rationalize difficult cases of VP coordination in a simple direct way.

References

William A. Baldwin and George O. Strawn. 1991. Multidimensional trees. *Theoretical Computer Science*, 84:293–311.

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

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

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

N. Chomsky and M. P. Schüzenberger. 1963. 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.

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

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

Roger Evans, Gerald Gazdar, and David Weir. 1995. 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.

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

Saul Gorn. 1967. 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.

The XTAG Research Group. 1998. A lexicalized tree adjoining grammar for english. Technical Report IRCS-98-18, Institute for Research in Cognitive Science.

Karin Harbusch, Friedbert Widmann, and Jens Woch. 1998. Towards a workbench for schema-TAGs. In Anne Abeillé, Tilman Becker, Owen Rambow, Giorgio Satta, and K. Vijay-Shanker, editors, *Fourth Interna-*

tional Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4), pages 58–61.

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

D. Terrence Langendoen. 1976. On the weak generative capacity of infinite grammars. *CUNYForum*, 1:13–24.

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

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

James Rogers. 1998a. A descriptive characterization of tree-adjoining languages (project note). 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. ACL. Project Note.

James Rogers. 1998b. The descriptive complexity of generalized local sets. In Uwe Moennich and Hans-Peter Kolb, editors, *The Mathematics of Syntactic Structure*. Mouton/deGruyter.

James Rogers. 1998c. 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.

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

Martine Smets. 1998. 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.

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

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

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

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

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

Fei Xia, Martha Palmer, K. Vijay-Shanker, and Joseph Rosenzweig. 1998. 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.