On the Degree of Nondeterminism of Tree Adjoining Languages and Head Grammar Languages

Suna Bensch, Maia Hoeberechts

On the Degree of Nondeterminism of Tree Adjoining Languages and Head Grammar Languages

Suna Bensch[1] and Maia Hoeberechts[2⋆]

[1] Department of Computing Science, Umeå University, Umeå, Sweden
suna@cs.umu.se
[2] Ocean Networks Canada and Department of Computer Science, University of Victoria, Victoria, Canada
maiah@uvic.ca

Abstract. The degree of nondeterminism is a measure of syntactic complexity which was investigated for parallel and sequential rewriting systems. In this paper, we consider the degree of nondeterminsm for tree adjoining grammars and their languages and head grammars and their languages. We show that a degree of nondeterminism of 2 suffices for both formalisms in order to generate all languages in their respective language families. Furthermore, we show that deterministic tree adjoining grammars (those with degree of nondeterminism equal to 1), can generate non-context-free languages, in contrast to deterministic head grammars which can only generate languages containing a single word.

Keywords: Tree Adjoining Languages, Head Grammar Languages, Degree of Nondeterminism

1 Introduction

The degree of nondeterminism for tabled Lindenmayer systems and languages has been studied in [9] and [8] as a measure of syntactic complexity. The degree of nondeterminism has also been considered for sequential rewriting systems in [2] and [3]. The degree of nondeterminism is usually defined as the maximal number of production rules with the same left-hand side which provides a measure of the amount of choice available during derivations using the grammar. In this paper we consider the degree of nondeterminism for tree adjoining grammars and head grammars. *Tree adjoining grammars* were first introduced in [5] and their formal properties and linguistic relevance have been considered in [4] and [10]. TAGs are tree-generating grammars which use an *adjoining* operation that generates new trees by joining and attaching two different trees at a particular node. *Head Grammars* were first introduced in [7]. The principle feature which distinguishes a Head Grammar (HG) from a context-free grammar is that the

⋆ This research has been supported in part by an NSERC scholarship and by NSERC grant OGP 249 (Helmut Jürgensen).

head grammar includes a wrapping operation which allows one string to be inserted into another string at a specific point (the head). It is known that for both tree adjoining grammars and head grammars, the class of string languages generated by the grammars is larger than the class of context-free languages (e.g. they are able to define the language $\{a^n b^n c^n d^n | n \geq 0\}$ [10]). In [10] it is shown that the two formalisms generate exactly the same class of string languages, and that these languages are *mildly context-sensitive*.

The notion of mild context-sensitivity tries to capture mathematical and computational properties that formal models for the description and analysis of natural language should possess. The notion of mild context-sensitivity was first mentioned in [4] and sparked active research yielding to many different approaches and definitions thereof (see, for example, [6]). There has been much discussion about the linguistic differences between mildly context-sensitive grammar formalisms, and in general, investigations mainly focus on polynomial parsing algorithms. Formal properties of mild context-sensitive grammar formalisms have not been as extensively considered. The examination of degree of nondeterminism for TAGs and MHGs is a step in that direction. It would be interesting to consider whether there are any linguistic implications for the degree of nondeterminism — for example, are there aspects of natural language modelling which are best done with a grammar having a higher (or lower) degree of nondeterminism than others?

2 Notational Conventions

The reader is assumed to be familiar with the basic notions in formal language theory. We use the following notational conventions and definitions in this paper. $|S|$ denotes the cardinality of the set S, $\emptyset$ denotes the empty set, $\cup$ denotes set union, and $\setminus$ denotes set difference. S is called an *alphabet* if it is a finite nonempty set of symbols. $\mathbb{N}$ denotes the set of natural numbers $\{1, 2, 3, \ldots\}$. For any set X, a *word* over X is a finite sequence of symbols from X. λ will be used to denote the empty word. The *concatenation* of two words x and y is denoted by xy and represents the word formed by the juxtaposition of x and y. The concatenation of a word x and a set S is $xS = \{xy \mid y \in S\}$ (Sx is similarly defined). X^* is the free monoid generated by X with concatenation as binary operation and λ as identity element. $X^+ = X^* \setminus \lambda$.

3 Tree Adjoining Grammars (TAGs)

Tree Adjoining Grammars (TAGs) are linguistically motivated tree-generating grammars which were originally introduced in [5]. For linguistic applications, TAGs have an advantage over string generating grammars such as context-free grammars because the elementary objects and all the objects generated are trees, which represent syntactic structure explicitly, as opposed to strings, which do not. In what follows, we give an informal description of TAGs first and their formal definition later.

The components of a TAG are a set of *initial trees* and a set of *auxiliary trees*. Each node in an initial tree or an auxiliary tree is labelled by a terminal symbol, by λ, or by a nonterminal symbol and constraints (which serve to restrict adjunction at that node, as will be explained later). The initial trees are the axioms used in the generation of new trees. The only means by which new trees are generated is the adjunction operation, which allows an auxiliary tree to be inserted into an initial tree or a derived (i.e. previously generated) tree. TAGs which include a second operation, substitution, are not discussed here as they are equivalent in generative power to TAGs which use only adjoining.

Tree adjunction is illustrated in Figure 1 (adapted from [10]). The tree shown on the left, γ, is an initial tree or a derived tree. The root node of γ is labelled by the nonterminal A, and γ contains an interior node n which is labelled by the nonterminal B. The tree in the centre, β, is an auxiliary tree in which both the root node and the *foot node*, a special node on the frontier of the tree, are labelled by B. As the nonterminal labelling n in γ and the nonterminal labelling the root node of β are the same, it is possible to adjoin β at n. Adjunction results in the new tree γ' which is constructed by removing the subtree rooted at n from γ, inserting β into γ at the point where n was removed, and then replacing the foot node in β by the subtree originally rooted at n.

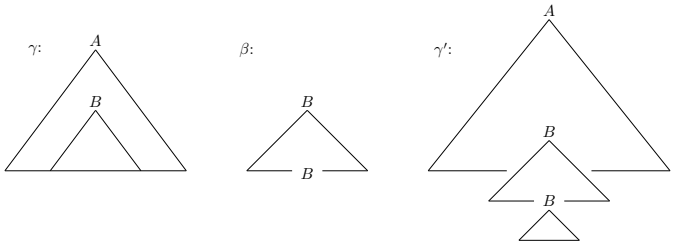

Fig. 1. Tree Adjunction

The language defined by a TAG G is the set of all words which are produced as the *yield* of some tree generated through zero or more adjunction operations in G. The yield of a tree is the word obtained by concatenating the terminal symbols on the leaf nodes of the tree, read from left to right. In a TAG, initial trees must have only terminal symbols on their leaf nodes, and auxiliary trees have terminal symbols on all leaf nodes except for the foot node. Thus, every tree generated through adjunction operations in G has a terminal word as its yield, and that word is an element of $L(G)$. All trees discussed in this paper are finite.

In a TAG, a node n in a tree γ is labelled either by a terminal symbol, by λ, or by a triple of the form $\langle A, sa_{(\gamma,n)}, oa_{(\gamma,n)}\rangle$ where A is a nonterminal symbol and

$sa_{(\gamma,n)}$ and $oa_{(\gamma,n)}$ are called the *adjunction constraints* and have the following interpretations[3]:

- $sa_{(\gamma,n)}$ is a set of trees. For a given node, $sa_{(\gamma,n)}$ (*selective adjunction*) is the set of trees from the grammar which are allowed to be adjoined at that node. We assume that for all β in $sa_{(\gamma,n)}$, the root node of β is labelled by the same nonterminal as n. If $sa_{(\gamma,n)} = \emptyset$, then adjunction is not permitted at that node, and we can write NA to indicate a *null adjunction* constraint.
- $oa_{(\gamma,n)} \in \{\mathsf{true}, \mathsf{false}\}$. If $oa_{(\gamma,n)}$ has the value true then we speak about an OA (*obligatory adjunction*) constraint and if $oa_{(\gamma,n)}$ has the value false then this indicates that adjunction is optional.

Definition 1. A *Tree Adjoining Grammar* (TAG) is a quadruple $G = (N, T, \mathcal{I}, \mathcal{A})$, where, N is the alphabet of nonterminal symbols, T the alphabet of terminal symbols, $N \cap T = \emptyset$, $\mathcal{I}$ and $\mathcal{A}$ are given as follows

- $\mathcal{I}$ is a finite set of *initial trees* where each $\alpha \in \mathcal{I}$ satisfies:
  - All interior nodes of α are labelled by $\langle B, sa_{(\alpha,n)}, oa_{(\alpha,n)}\rangle$, $B \in N$, $sa_{(\alpha,n)} \subseteq \mathcal{A}$ and $oa_{(\alpha,n)} \in \{\mathsf{true}, \mathsf{false}\}$.
  - All leaf nodes of α are labelled by some $u \in \{T \cup \{\lambda\}\}$.
- $\mathcal{A}$ is a finite set of *auxiliary trees* where each $\beta \in \mathcal{A}$ satisfies:
  - All interior nodes of β are labelled by $\langle B, sa_{(\beta,n)}, oa_{(\beta,n)}\rangle$, $B \in N$, $sa_{(\beta,n)} \subseteq \mathcal{A}$ and $oa_{(\beta,n)} \in \{\mathsf{true}, \mathsf{false}\}$.
  - All leaf nodes of β are labelled by some $u \in \{T \cup \{\lambda\}\}$, except the foot node, denoted by $ft(\beta)$, which carries the same category (but not necessarily the same adjunction constraints) as the root node.

Tree adjunction is a partial ternary operation $\nabla(\gamma, \beta, n)$ which produces a new tree, γ', which is a copy of γ with the auxiliary tree β inserted at the node with address n. We define a *derived tree* to be an initial tree, an auxiliary tree or a tree produced by an application of ∇. We say adjunction is *permitted* when the following conditions hold for the arguments of ∇: γ is a derived tree, β is an auxiliary tree, and n is the address of an interior node in γ with label $\gamma(n) = \langle B, sa_{(\gamma,n)}, oa_{(\gamma,n)}\rangle$. The root node of β must be labelled by the same nonterminal as n, that is, $\langle B, sa_{(\beta,\lambda)}, oa_{(\beta,\lambda)}\rangle$, and β must be an element of $sa_{(\gamma,n)}$.

After adjunction the labels of the nodes are unchanged from their original labels in γ and β, except for the nodes affected by adjunction: the node at address n which now carries the label from the root node of β, and the foot node of β with the change that the *oa* constraint on the node is set to false.

For a TAG $G = (N, T, \mathcal{I}, \mathcal{A})$, a derivation in G will be denoted by $\underset{G}{\Longrightarrow}$. The tree γ' can be derived from γ if and only if there exist $\beta \in \mathcal{A}$ and n in *gamma* such that adjunction of β in γ at n is permitted and $\nabla(\gamma, \beta, n) = \gamma'$. Then we write $\gamma \underset{G}{\Longrightarrow} \gamma'$. Let $\overset{*}{\underset{G}{\Longrightarrow}}$ denote the reflexive, transitive closure of $\underset{G}{\Longrightarrow}$.

[3] This notation has been changed slightly from [10] to include an index on the *sa* and *oa* constraint.

The *tree language* generated by G is the set of all trees which can be generated in zero or more derivation steps from the initial trees of G, and in which no nodes remain which are labelled by an OA (obligatory adjunction) constraint.

$$T(G) = \{\gamma \mid \alpha \overset{*}{\underset{G}{\Longrightarrow}} \gamma \text{ for some } \alpha \in \mathcal{I} \text{ and } \gamma \text{ has no OA nodes } \}$$

The *yield* of a tree is the string one obtains by concatenating the labels on the leaf nodes from left to right.

For a TAG $G = (N, T, \mathcal{I}, \mathcal{A})$ with tree language $T(G)$, the tree adjoining language generated by G is

$$L(G) = \{yield(\gamma) \mid \gamma \in T(G)\}$$

Let $\mathcal{L}^{\text{TAG}}$ represent the family of tree adjoining languages.

3.1 Degree of Nondeterminism for TAGs

The degree of nondeterminism for tree adjoining grammars will measure the amount of choice between auxiliary trees which can be adjoined within a given TAG. When defining the degree of nondeterminism for TAGs an essential ambiguity in the interpretation has to be taken into account. On the one hand, when defining the degree of nondeterminism for a given node n in a tree γ, one could consider only the auxiliary trees in the set $sa_{(\gamma,n)}$, which can be adjoined at that node. On the other hand, one could consider *all* auxiliary trees in the set $\mathcal{A}$ for the given tree adjoining grammar (even if they are not in the set $sa_{(\gamma,n)}$). We will call these views *weak degree of nondeterminism* and *strong degree of nondeterminism*, respectively. In this section we will define strong and weak degree of nondeterminism, and then show that the two measures are equivalent. For the following definitions, consider a TAG $G = (N, T, \mathcal{I}, \mathcal{A})$. Let γ represent an arbitrary tree in $\mathcal{I} \cup \mathcal{A}$ and $\beta \in \mathcal{A}$ represent an arbitrary auxiliary tree.

Definition 2. Weak degree of nondeterminism

- For a node in γ at address n labelled by $\langle B, sa_{(\gamma,n)}, oa_{(\gamma,n)}\rangle$, the *degree of the node* is denoted by $Deg_G(\gamma, n)$, and is defined as the number of trees in the selective adjunction set for the node. That is, $Deg_G(\gamma, n) = |sa_{(\gamma,n)}|$.
- The *weak degree of nondeterminism of a tree adjoining grammar* G is denoted by $Det_w(G)$, and is defined as the maximal degree of any node in a tree in G: $Det_w(G) = \max\{Deg_G(\gamma, n) \mid \gamma \in \mathcal{I} \cup \mathcal{A}, n \in dom(\gamma)\}$.
- The *weak degree of nondeterminism of a tree adjoining language* L, $Det_w(L)$, is defined as the minimal weak degree of nondeterminism of any TAG capable of generating L: $Det_w(L) = \min\{Det_w(G) \mid G \text{ is a TAG with } L(G) = L\}$.

Definition 3. Strong degree of nondeterminism

- The *degree of a nonterminal* $B \in N$, denoted by $Deg_G(B)$, is the number of auxiliary trees in $\mathcal{A}$ which have B labelling their root node: $Deg_G(B) = |\{\beta \mid \beta \in \mathcal{A}, \beta(\lambda) = \langle B, sa_{(\beta,\lambda)}, oa_{(\beta,\lambda)}\}|$.

- The *strong degree of nondeterminism for a tree adjoining grammar* G, denoted by $Det_s(G)$, is defined as the maximal degree of a nonterminal in N: $Deg_s(G) = \max\{Deg_G(B) \mid B \in N\}$.
- The *strong degree of nondeterminism of a tree adjoining language* L, $Det_s(L)$, is defined as the minimal strong degree of nondeterminism of any TAG capable of generating L: $Det_s(L) = \min\{Det_s(G) \mid G \text{ is a TAG with } L(G) = L\}$.

We will now show that strong and weak degree of nondeterminism are equivalent measures for TAGs.

Theorem 1. Given a TAG $G = (N, T, \mathcal{I}, \mathcal{A})$, $Det_w(G) \leq Det_s(G)$.

Proof. By definition, for every node n in a tree $\gamma \in \mathcal{I} \cup \mathcal{A}$ labelled by $\langle B, sa_{(\gamma,n)}, oa_{(\gamma,n)}\rangle$, $sa_{(\gamma,n)} \subseteq \mathcal{A}$. Therefore, $Deg_G(\gamma, n) \leq Deg_G(B)$ for any given node labelled by B in a tree γ at n, and thus $Det_w(G) \leq Det_s(G)$.

Theorem 2. Given a TAG $G = (N, T, \mathcal{I}, \mathcal{A})$ with $Det_w(G) < Det_s(G)$, there effectively exists a TAG $G' = (N', T', \mathcal{I}', \mathcal{A}')$ for which $L(G) = L(G')$ and $Det_w(G') = Det_s(G') = Det_w(G)$.

Proof. The intuitive idea behind the proof is that the set of auxiliary trees which can be adjoined at any given node is determined by two conditions: (i) the nonterminal symbol labelling the node, and (ii) the *sa* constraint which restricts the subset of auxiliary trees which are actually permitted to be adjoined at that node. The Algorithm 1 below works by making copies of the auxiliary trees such that for an *sa* set containing $\{\beta_1, \ldots, \beta_k\}$, k new auxiliary trees are introduced, whose root nodes are labelled by a common nonterminal which is used only for the auxiliary trees in that *sa* set. The result of copying and relabelling is that the strong degree of nondeterminism for the grammar is reduced to the weak degree because the number of auxiliary trees labelled by any given nonterminal is equal to the size of the *sa* set in which the auxiliary trees bearing that nonterminal appear. The algorithm recursively relabels all new auxiliary trees which are created.

An example of the effect of Algorithm 1 for one node is shown in Figure 2. The trees at the top, α_1, β_1 and β_2, are the trees from the TAG before relabelling takes place. In the new initial tree, α_1', the relabelled node can be seen. The new auxiliary trees, δ_1 and δ_2, are copies of β_1 and β_2 respectively for which new root and foot nodes have been added, and relabelling has been recursively applied to produce β_1' and β_2'.

Algorithm 1.
Preconditions: $G = (N, T, \mathcal{I}, \mathcal{A})$ is the TAG which will be relabelled.
Postconditions: A new TAG $G' = (N', T', \mathcal{I}', \mathcal{A}')$ is produced for which $L(G) = L(G')$ and $Det_w(G') = Det_s(G') = Det_w(G)$.
Let $N' = \emptyset$, $T' = T$, $\mathcal{I}' = \emptyset$, $\mathcal{A}' = \emptyset$
For each t in $\mathcal{I}$
  Let t' be a new tree name, $t' \notin (\mathcal{I} \cup \mathcal{A} \cup \mathcal{I}' \cup \mathcal{A}')$
  $t' = Relabel(N', t, \mathcal{A}')$
  Let $\mathcal{I}' = \mathcal{I}' \cup t'$
Function $Relabel(N', t, \mathcal{A}')$
Preconditions: N' is the set of nonterminal symbols defined so far, t is the tree currently being

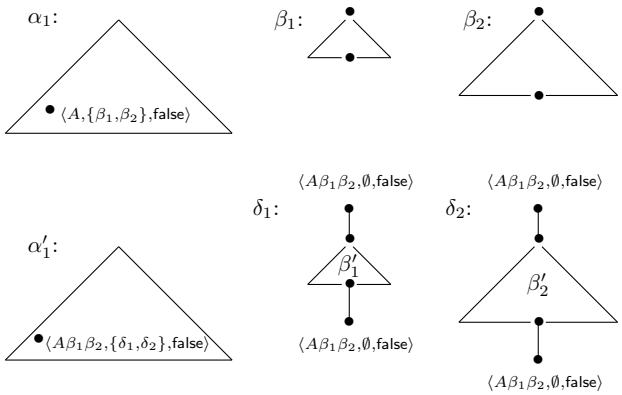

Fig. 2. Effect of Algorithm 1 for one node in α_1

considered, $\mathcal{A}'$ is the set of new auxiliary trees constructed so far
Postconditions: N' has been updated to include any new nonterminals, t is unchanged, $\mathcal{A}'$ has been updated to include any new trees resulting from the relabelling of t
Returns a new tree t' which is the relabelled t
Let t' be a copy of t
For each node n of t labelled by $\langle A, sa_{(t,n)}, oa_{(t,n)}\rangle$, with $sa_{(t,n)} = \{\beta_1, \ldots, \beta_k\}$
Let $A\beta_1 \cdots \beta_k$ be a nonterminal symbol
If $A\beta_1 \cdots \beta_k \notin N'$
Let $N' = N' \cup A\beta_1 \cdots \beta_k$
For each β_i in $sa_{(t,n)}$
Let δ_i be a new tree name, $\delta_i \notin \mathcal{A}'$
Let δ_i be an auxiliary tree constructed as follows:
- label the root node of δ_i by $\langle A\beta_1 \cdots \beta_k, \emptyset, \mathsf{false}\rangle$
- connect the root node of δ_i to the root node of a copy of β_i
- connect the foot node of the copy of β_i to the foot node of δ_i
- label the foot node of δ_i by $\langle A\beta_1 \cdots \beta_k, \emptyset, \mathsf{false}\rangle$

Let $\mathcal{A}' = \mathcal{A}' \cup \delta_i$
For i from 1 to k
$\delta_i = Relabel(N', \delta_i, \mathcal{A}')$
If $sa_{(t,n)} = \emptyset$
Let $sa'_{(t,n)} = \emptyset$
Else
Let $sa'_{(t,n)} = \{\tau \mid \tau \in A', \tau(\lambda) = \langle A\beta_1 \cdots \beta_k, sa_{(\tau,\lambda)}, oa_{(\tau,\lambda)}\rangle\}$
($sa'_{(t,n)}$ is the set of all trees in $\mathcal{A}'$ whose root nodes are labelled by $A\beta_1 \cdots \beta_k$)
Let the node corresponding to n in t' be labelled by $\langle A\beta_1 \cdots \beta_k, sa'_{(t,n)}, oa_{(t,n)}\rangle$
End Function

Thus, as the strong degree of nondeterminism can be reduced to the weak degree for any given TAG, one measure of degree of nondeterminism is sufficient. We can omit reference to strong or weak in our notation, and therefore, $Det(G)$ will be used to denote the (weak) degree of nondeterminism for TAGs, and $Det_{\mathrm{TAG}}(L)$ will denote the degree of nondeterminism for tree adjoining languages. Finally, we will show that for a TAG G with degree of nondeterminism greater than 2, we can create an equivalent TAG G' with degree of nondeterminism equal to 2. Thus, the degree of nondeterminism for any tree adjoining language is at most 2.

Theorem 3. Given a TAG $G = (N, T, \mathcal{I}, \mathcal{A})$ with $Det(G) > 2$, there effectively exists an TAG G' for which $L(G) = L(G')$ and $Det(G') = 2$.

Proof. The following Algorithm 2 examines all the nodes in the initial trees and auxiliary trees of G. When a node n in a tree γ is found with $Deg(\gamma, n) > 2$, this indicates that there is a choice between more than two auxiliary trees for adjunction at that node. Suppose the selective adjunction set for the node is $sa_{(\gamma,n)} = \{\beta_1, \ldots, \beta_n\}$ with $n > 2$. The algorithm works by introducing new auxiliary trees $\delta_1, \ldots, \delta_{n-2}$ each consisting of only a root node and foot node. The purpose of the δ_i trees is to reduce the choice between auxiliary trees to 2 at any given node. Node n is relabelled such that only β_1 or δ_1 can be adjoined, that is, $sa_{(\gamma,n)} = \{\beta_1, \delta_1\}$. At the root node of δ_1, β_2 or δ_2 can be adjoined, that is, $sa_{(\delta_1,\lambda)} = \{\beta_2, \delta_2\}$. Generally, for δ_i with $1 \leq i < n-2$, $sa_{(\delta_i,\lambda)} = \{\delta_{i+1}, \beta_{i+1}\}$ For δ_{n-2}, the root node is labelled by the sa set $sa_{\delta_{n-2},\lambda} = \{\beta_{n-1}, \beta_n\}$. Introduction of new auxiliary trees and relabelling is done for all nodes in G with degree greater than 2. The resulting TAG G' generates the same language as G, but contains no node with more than 2 trees in its sa set, and therefore $Det(G') = 2$.

Algorithm 2.
Preconditions: $G = (N, T, \mathcal{I}, \mathcal{A})$ is a TAG with $Det(G) > 2$
Postconditions: G has been modified such that $L(G)$ is unchanged and $Det(G) = 2$
For each $\gamma \in \mathcal{I} \cup \mathcal{A}$
  For each n in $dom(\gamma)$ labelled by $\langle A, sa_{(\gamma,n)}, oa_{(\gamma,n)}\rangle$ with $sa_{(\gamma,n)} = \{\beta_1, \ldots, \beta_k\}$
    If $k > 2$
      For i from 1 to $(k-2)$
        Let δ_i be a new tree name $\delta_i \notin (\mathcal{I} \cup \mathcal{A})$
        Let δ_i be an auxiliary tree constructed as follows:
          • δ_i consists of two nodes: the root node and the foot node
          • **If** $i < k-2$
            the root and foot are labelled by $\langle A, \{\beta_{i+1}, \delta_{i+1}\}, oa_{(\gamma,n)}\rangle$
            Else
            the root and foot are labelled by $\langle A, \{\beta_{i+1}, \beta_{i+2}\}, oa_{(\gamma,n)}\rangle$
        Let $\mathcal{A} = \mathcal{A} \cup \delta_i$
      Let the label of n be replaced by $\langle A, \{\beta_1, \delta_1\}, oa_{(\gamma,n)}\rangle$

Corollary 1. For any $L \in \mathcal{L}^{\text{TAG}}$, $Det_{\text{TAG}}(L) \leq 2$.

4 Modified Head Grammars (MHGs)

We will consider Modified Head Grammars (MHGs) which were proposed in [11], and differ only slightly from the definition in [7]. The strings used in modified head grammars are called *headed strings*. In a headed string, a special position between two symbols, marked by ↑, is designated as the *head* of the string. MHGs use a wrapping operation to insert one string into another, and the purpose of the head is to designate the insertion point during this operation. For an alphabet X, let $\mathcal{H}_X$ be the set of headed strings over X. $\mathcal{H}_X$ is defined as:

$$\mathcal{H}_X = \{v{\uparrow}w \mid v, w \in X^*\}$$

For example, for the alphabet $X = \{a, b, c\}$, $abc{\uparrow}cbacba$, $\lambda{\uparrow}aaa$ and $\lambda{\uparrow}\lambda$ are three of the elements of $\mathcal{H}_X$.

The production rules of an MHG are defined in terms of two types of operations, wrapping and concatenation, which are performed on headed strings.

The *wrapping operation*, $W : \mathcal{H}_X^2 \to \mathcal{H}_X$ is a binary operation which has the effect of inserting one string into another at the head. Given headed strings $v_1{\uparrow}w_1$ and $v_2{\uparrow}w_2$, the result of applying W is a new headed string comprised of $v_2{\uparrow}w_2$ inserted into $v_1{\uparrow}w_1$ at its head:

$$W(v_1{\uparrow}w_1, v_2{\uparrow}w_2) = v_1v_2{\uparrow}w_2w_1$$

The *concatenation* of headed strings is an n-ary operation denoted by $C_{m,n} : \mathcal{H}_X^n \to \mathcal{H}_X$, where n is the number of headed strings to be concatenated and m is the index of the string whose head becomes the head for the resulting string. The indices must satisfy $n \geq 1$ and $1 \leq m \leq n$. The interpretation of $C_{m,n}$ is as follows:

$$C_{m,n}(v_1{\uparrow}w_1, v_2{\uparrow}w_2, \ldots, v_m{\uparrow}w_m, \ldots, v_n{\uparrow}w_n) = v_1w_1v_2w_2 \cdots v_m{\uparrow}w_m \cdots v_nw_n$$

Given a nonterminal alphabet N and a terminal alphabet T, a *headed string expression* over N and T is recursively defined as follows:

- Every headed string $\sigma \in \mathcal{H}_T$ is a headed string expression.
- For all $A \in N$, A is a headed string expression.
- If σ_1 and σ_2 are headed strings expressions, then $W(\sigma_1, \sigma_2)$ is a headed string expression.
- If $\sigma_1, \ldots, \sigma_n$ are headed string expressions, then $C_{i,n}(\sigma_1, \ldots, \sigma_n)$ are headed string expressions for $1 \leq i \leq n$.
- There are no other headed string expressions.

Let $\mathcal{E}_{N,T}$ represent the set of headed string expressions over N and T. By convention, we will use σ to represent a headed string expression. If a headed string expression contains no nonterminals, we call it *closed*.

Definition 4. A *Modified Head Grammar (MHG)* is a quadruple, $G = (N, T, P, S)$ where N is a finite set of nonterminal symbols, T is a finite set of terminal symbols, $S \in N$ is the start symbol, P is a set of production rules $\{p_1, \ldots, p_k\}$, where $p_i = A \to \sigma_i$, with $A \in N, \sigma_i \in \mathcal{E}_{N,T}$.

Consider an MHG, $G = (N, T, P, S)$, with $p_i = A \to \sigma_i \in P$. Given a headed string expression $\sigma \in \mathcal{E}_{N,T}$ containing a nonterminal A, we may apply the rule p_i to replace one instance of A in σ by the right hand side of p_i, σ_i. Let σ' denote the resulting string. Then we write $\sigma \underset{G}{\Longrightarrow} \sigma'$ to indicate that σ' can be derived from σ using a production rule in G. If the grammar in use is clear from the context, we write $\Longrightarrow$ rather than $\underset{G}{\Longrightarrow}$. Let $\overset{*}{\underset{G}{\Longrightarrow}}$ denote the reflexive, transitive closure of $\underset{G}{\Longrightarrow}$.

For an MHG $G = (N, T, P, S)$, the *expression language* generated by G, $E(G)$, is the set of all closed headed string expressions which can be derived from S using the rules of G. Formally,

$$E(G) = \{\sigma \mid S \overset{*}{\underset{G}{\Longrightarrow}} \sigma, \sigma \text{ is closed}\}$$

The *head language* generated by G, $H(G)$, is the set of headed strings which result from the evaluation according to the definitions of W and $C_{m,n}$ of the closed headed string expressions in $E(G)$:

$$H(G) = \{v{\uparrow}w \mid v{\uparrow}w \in E(G)\}$$

The *language generated by* G, $L(G)$, is the set of strings one obtains by removing the heads from the strings in $H(G)$:

$$L(G) = \{vw \mid v{\uparrow}w \in H(G)\}$$

Let $\mathcal{L}^{\mathrm{MHG}}$ denote the family of languages which can be defined by MHGs.

4.1 Degree of Nondeterminism for MHGs

Let $G = (N, T, P, S)$ be an MHG. For a nonterminal A, let P_A be the set of production rules with A on the left-hand side.
That is, $P_A = \{p_i \mid p_i \in P, p_i = A \to \sigma\}$.

Definition 5. Degree of nondeterminism

- The *degree of the nonterminal* A, denoted by $Deg_G(A)$, is the number of production rules with A on the left-hand side: $Deg_G(A) = |P_A|$
- The *degree of nondeterminism* of the MHG G, $Det(G)$, is defined as the maximum degree of a nonterminal in G: $Det(G) = \max\{Deg_G(A) \mid A \in N\}$. Intuitively, the degree of nondeterminism measures how much choice between productions rules there is during a derivation using a specific MHG.
- $Det_{\mathrm{MHG}}(L)$, the *degree of nondeterminism for an MHG language* L, is defined as the minimal degree of nondeterminism of any MHG capable of generating L: $Det_{\mathrm{MHG}}(L) = \min\{Det(G) \mid G \text{ is an MHG for which } L(G) = L\}$.

It will now be shown that the degree of nondeterminism for any MHG language is at most 2. The proof of Theorem 4 contains an algorithm which generates an MHG with degree of nondeterminism equal to 2 from any MHG with degree of nondeterminism greater than 2.

Theorem 4. Given an MHG $G = (N, T, P, S)$ with $Det(G) > 2$, there effectively exists an MHG $G' = (N', T, P', S')$ for which $L(G') = L(G)$ and $Det(G') = 2$.

Proof. An MHG G with $Det(G) > 2$ contains nonterminals $A \in N$ which appear on the left hand side of more than 2 production rules. The Algorithm 3 presented below introduces new nonterminal symbols and production rules so that the choice between production rules at any point in a derivation is always binary. To understand how it works, suppose there is a nonterminal $A \in N$ which appears on the left hand side of 3 rules, $p_1 : A \to \sigma_1$, $p_2 : A \to \sigma_2$ and $p_3 : A \to \sigma_3$. After execution of the algorithm, the nonterminal A would be replaced by three nonterminal, A_1,A_2 and A_3, and the rules p_1,p_2 and p_3 would be replaced by six

production rules: $p'_1 = A_1 \to \sigma'_1$, $p''_1 = A_1 \to A_2$, $p'_2 = A_2 \to \sigma'_2$, $p''_2 = A_2 \to A_3$, $p'_3 = A_3 \to \sigma'_3$, $p''_3 = A_3 \to A_1$. The rules are "chained" such that the same strings can be derived but the choice of production rules at any given time is reduced to 2.

Algorithm 3.
Preconditions: $G = (N, T, P, S)$ is an MHG with $Det(G) > 2$
Postconditions: New MHG $G' = (N', T', P', S')$ such that $L(G') = L(G)$ and $Det(G') = 2$
Let $N' = \emptyset$, $T' = T$, $P' = \emptyset, S' = S_1$
For each $A \in N$
  Let $P_A = \{p_1, \ldots p_j\}$ be the set of rules with A on the left hand side
  Let A_1 be a new nonterminal, $A_1 \notin (N \cup N')$
  Let $N' = N' \cup A_1$
  For $i = 1$ **to** j
    From $p_i = A \to \sigma_i$
    Create rule $p'_i = A_i \to \sigma'_i$
      where σ'_i is the headed string expression which results
      by replacing all $B \in N$ appearing in σ_i by B_1
    Let $P' = P' \cup p'_i$
    If $i < j$
      Let A_{i+1} be a new nonterminal, $A_{i+1} \notin (N \cup N')$
      Create rule $p''_i = A_i \to A_{i+1}$
      Let $N' = N' \cup A_{i+1}$, $P' = P' \cup p''_i$
    Else If $j > 1$
      Create rule $p''_i = A_i \to A_1$
      Let $P' = P' \cup p''_i$

Corollary 2. For any $L \in \mathcal{L}^{\text{MHG}}$, $Det_{\text{MHG}}(L) \leq 2$.

A *deterministic MHG* is an MHG $G = (N, T, P, S)$ for which $Det(G) = 1$. In other words, no nonterminal $A \in N$ appears on the left-hand side of more than one production rule $p_i \in P$.

Theorem 5. A deterministic MHG $G = (N, T, P, S)$ defines a language with $|L(G)| \leq 1$.

Proof. We can observe the following requirements for the production rules of a deterministic MHG with a nonempty language: (i) At least one production rule must have only terminal symbols or λ on the right-hand side. (ii) The same nonterminal symbol may not appear on the left and right-hand side of a given production rule. (iii) There can be no set of production rules $P_{\text{cycle}} \subseteq P = p_1, \ldots, p_k$ which have the following form: $p_1 : A_1 \to \sigma_1$ where σ_1 contains $A_1 \ldots$ $p_i : A_i \to \sigma_i$, where σ_i contains $A_{i+1} \ldots p_k : A_k \to \sigma_k$, where σ_k contains A_1.

(i) is necessary so that it is possible to derive a headed string expression which is closed. (ii) and (iii) are necessary so that the sequence of derivations does not contain a loop. Such a loop would prevent the sequence of derivations from ending since each nonterminal appears on the left hand side of only one rule, and therefore the derivation leading to the loop would be chosen every time. Thus, since the sequence of derivations does not contain a loop and must start from S, if $L(G)$ is nonempty then $|L(G)| = 1$.

5 Conclusions

The relationship between TAGs and MHGs was explored in several papers [10] and [11]. In this paper, we have shown that for both TAGs and MHGs, the de-

gree of nondeterminism 2 suffices to generate all languages in their respective language families. Reducing the degree of nondeterminism with our algorithms can increase the number of elementary trees in a TAG or the number of production in a MHG considerably. We note that there is a significant difference between deterministic MHGs and deterministic TAGs. In [10], an example of a TAG appears which has only one auxiliary tree (and is therefore deterministic by our definition), and yet it generates the language $\{a^n b^n c^n d^n \mid n \geq 0\}$ which is noncontext-free. By contrast, deterministic MHGs are only capable of generating languages for which $|L| \leq 1$. Finally, there is a small question which arose concerning TAGs during our work. Although we know that deterministic TAGs are capable of generating noncontext-free languages, we did not identify the class of languages which can be generated by deterministic TAGs.

Acknowledgements

We thank Henning Bordihn for helping us to understand and define Degree of Nondeterminism for TAGs. Thank you to Helmut Jürgensen for his comments on drafts of this paper and for his support and encouragement for this project.

References

1. Aydin, S., Bordihn, H.: Sequential Versus Parallel Grammar Formalisms with Respect to Measures of Descriptional Complexity. Fundamenta Informaticae 55, 243–254 (2003)
2. Bordihn, H.: Über den Determiniertheitsgrad reiner Versionen formaler Sprachen. PhD thesis, Technische Universität "Otto von Guericke" Magdeburg (1992)
3. Bordihn, H.: On the degree of nondeterminism. In *Developments in Theoretical Computer Science*, J. Dassow and A. Kelemenova, Eds., Gordon and Breach Science Publishers, 133–140 (1994)
4. Joshi, A.: Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In *Natural Language Parsing*, D. Dowty and L. Karttunen and A. Zwicky, Eds., Cambridge University Press, 206–250 (1985)
5. Joshi, A., Levi, L.S., Takahashi, M.: Tree adjunct grammars. Journal of Computer and System Sciences 10, 136–163 (1975)
6. Marcus, S.: Mild Context-Sensitivity, After Twenty Years. Fundamenta Informaticae 73, 203–204 (2006)
7. Pollard, C.: Generalized Phrase Structure Grammars, Head Grammars and Natural Language. PhD thesis, Stanford University (1984)
8. Rozenberg, G.: Extension of tabled 0L systems and languages. International Journal of Computer and Information Sciences 2, 311–336 (1973)
9. Rozenberg, G.: T0L systems and languages. Information and Control 23, 357–381 (1973)
10. Vijay-Shanker, K., Weir, D.: The equivalence of four extensions of context-free grammars. Mathematical Systems Theory 87, 511–546 (1994)
11. Weir, D., Vijay-Shanker, K., Joshi, A.: The relationship between tree adjoining grammars and head grammars. In *Proceedings of the 24th Annual Meeting of Computational Linguistics, New York, NY* (1986)