US
UNIVERSITY OF SUSSEX

Sussex Research

The equivalence of four extensions of context-free grammars

K. Vijay-Shanker, David Weir

Publication date
01-01-1994

Licence
This work is made available under the Copyright not evaluated licence and should only be used in accordance with that licence. For more information on the specific terms, consult the repository record for this item.

Citation for this work (American Psychological Association 7th edition)
Vijay-Shanker, K., & Weir, D. (1994). *The equivalence of four extensions of context-free grammars* (Version 1). University of Sussex. https://hdl.handle.net/10779/uos.23310593.v1

Published in
Mathematical Systems Theory / Theory of Computer Systems

Link to external publisher version
https://doi.org/10.1007/BF01191624

Copyright and reuse:
This work was downloaded from Sussex Research Open (SRO). This document is made available in line with publisher policy and may differ from the published version. Please cite the published version where possible. Copyright and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners unless otherwise stated. For more information on this work, SRO or to report an issue, you can contact the repository administrators at sro@sussex.ac.uk. Discover more of the University's research at https://sussex.figshare.com/

THE EQUIVALENCE OF FOUR EXTENSIONS OF CONTEXT-FREE GRAMMARS*

K. Vijay-Shanker
Department of Computer and Information Science
University of Delaware
Newark, DE 19716

David J. Weir
School of Cognitive and Computing Sciences
University of Sussex
Brighton, BN1 9QH

January 25, 1994

Abstract

There is currently considerable interest among computational linguists in grammatical formalisms with highly restricted generative power. This paper concerns the relationship between the class of string languages generated by several such formalisms viz. Combinatory Categorial Grammars, Head Grammars, Linear Indexed Grammars and Tree Adjoining Grammars. Each of these formalisms is known to generate a larger class of languages than Context-Free Grammars. The four formalisms under consideration were developed independently and appear superficially to be quite different from one another. The result presented in this paper is that all four of the formalisms under consideration generate exactly the same class of string languages.

1 Introduction

There is currently considerable interest among computational linguists in grammatical formalisms with highly restricted generative power. This is based on the argument that a grammar formalism should not merely be viewed as a notation, but as part of the linguistic theory [6]. It should make predictions about the structure of natural language and its value is lessened to the extent that it supports both good and bad analyses. In order for a grammar

*This work has been supported by NSF grants MCS-82-19116-CER, MCS-82-07294, DCR-84-10413, IRI-8909810, ARO grant DAA29-84-9-0027, and DARPA grant N0014-85-K0018. The authors would like to thank Aravind Joshi and Mark Steedman for their important contributions to the research that led to this paper. We are also extremely grateful to Joost Engelfriet and the anonymous reviewers for their helpful critiques of earlier versions of the paper.

formalism to have such predictive power its generative capacity must be constrained. This has led to interest in the use of context-free grammars (cfg) [16, 6] as a notation with which to express linguistic theories. However, it is now generally accepted that cfg lack the generative power needed for this purpose [19, 3]. As a result there is substantial interest in the development and study of constrained grammar formalisms whose generative power exceeds cfg. This paper concerns the relationship between several of the most widely studied such formalisms.

We compare the class of languages that are generated by combinatory categorial grammars, head grammars, linear indexed grammars and tree adjoining grammars. Each of these formalisms is known to generate a larger class of languages than cfg. Furthermore, this class includes the non-context-free languages that have been used as the basis for the most widely accepted arguments that generative power greater than that of cfg is needed to generate certain natural languages [19, 3].

The four formalisms under consideration were developed independently and superficially differ considerably from one another. Informally, differences between the formalism can be explained in terms of the way in which they can be seen to extend cfg. For example, in addition to string concatenation, head grammars involve a wrapping operation with which one pair of strings can be wrapped around a second pair. In other respects head grammars are identical to cfg since the derivation process involves context-free rewriting of members of a finite set of nonterminal symbols. Both combinatory categorial grammars and linear indexed grammars, on the other hand, involve only string concatenation. However, they differ from cfg in that their derivation process involves rewriting of unbounded stack-like structures. The status of tree adjoining grammars, a tree manipulating system, is ambiguous since it is possible to interpret tree adjoining grammars as extending cfg in either of these ways.

Despite these differences, evidence existed suggesting that the weak generative capacity of these formalisms may be closely related. It appeared that while each of these formalisms had greater power than cfg, this extra power was very limited. In addition each formalism was apparently limited to a similar extent. This evidence involved the lack of an example of a language that could be generated by one formalism and not another. The languages $\{ ww \mid w \in \{ a, b \}^* \}$, $\left\{ ww^R ww^R \mid w \in \{ a, b \}^* \right\}$ [1] and $\{ a^n b^n c^n d^n \mid n \geq 0 \}$ are examples of languages that were known to be generated by all four formalisms, whereas, the languages $\{ www \mid w \in \{ a, b \}^* \}$ and $\{ a^n b^n c^n d^n e^n \mid n \geq 0 \}$ are not generated by any of the formalisms [17, 22].

The result presented in this paper is that all four of the formalisms under consideration generate exactly the same class of string languages. Preliminary versions of some of these results have appeared in [25, 28]. In Section 2 we define each of the formalisms and give the equivalence proofs in Section 3.

[1] w^R is the reverse of w.

2 Definitions of Formalisms

Three of the four formalisms (tree adjoining grammars, head grammars, and combinatory categorial grammars) have been used as the notation underlying various linguistic theories and linear indexed grammars have been discussed in [5] in connection with the relevance of indexed grammars to natural language. We will not discuss these theories here, however, as we define each formalism we will refer to relevant linguistic work.

2.1 Head Grammars

Head grammars (hg) were introduced in [15] where their linguistic relevance was investigated. Some of their formal properties were studied in [17]. They can be viewed as a generalization of cfg in which a wrapping operation is used in addition to concatenation. The nonterminals of a cfg derive strings of terminals. The nonterminals of a hg derive headed strings or pairs of terminal strings (u, v) that we denote $u_{\uparrow}v$. In Pollard's original (equivalent) definition [15], headed strings involved the additional specification that either the last terminal symbol in u or the first terminal symbol in v was the head. We find it mathematically cleaner to view a headed string simply as a pair since in Pollard's notation the empty headed string was rather problematic since it contained no head.

Definition 2.1 A hg is a four tuple $G = (V_N, V_T, S, P)$ where
V_N is a finite set of nonterminal symbols,
V_T is a finite set of terminal symbols,
$S \in V_N$ is the start symbol and
P is a finite set of productions of the form $A \rightarrow f(\sigma_1, \ldots, \sigma_n)$
where $A \in V_N$, $n \geq 1$, $f \in \{ W, C_{1,n}, C_{2,n}, \ldots, C_{n,n} \}$, $\sigma_1, \ldots, \sigma_n \in V_N \cup (V_T^* \times V_T^*)$,
and if $f = W$ then $n = 2$.

$C_{i,n} : (V_T^* \times V_T^*)^n \rightarrow (V_T^* \times V_T^*)$ is a concatenation operation where

$$C_{i,n}(u_{1\uparrow}v_1, \ldots, u_{i\uparrow}v_i, \ldots, u_{n\uparrow}v_n) = u_1v_1 \ldots u_{i\uparrow}v_i \ldots u_nv_n$$

$W : (V_T^* \times V_T^*)^2 \rightarrow (V_T^* \times V_T^*)$ is the wrapping operation where

$$W(u_{1\uparrow}v_1, u_{2\uparrow}v_2) = u_1u_{2\uparrow}v_2v_1$$

Given a hg, $G = (V_N, V_T, S, P)$, $\underset{G}{\Longrightarrow}$ is defined as follows.

- $u_{\uparrow}v \overset{0}{\underset{G}{\Longrightarrow}} u_{\uparrow}v$ for all $u_{\uparrow}v \in V_T^* \times V_T^*$.
- If $A \rightarrow f(\sigma_1, \ldots, \sigma_n) \in P$ then

$$A \overset{k}{\underset{G}{\Longrightarrow}} f(u_{1\uparrow}v_1, \ldots, u_{n\uparrow}v_n)$$

where $\sigma_i \overset{k_i}{\underset{G}{\Longrightarrow}} u_{i\uparrow}v_i$ $(1 \leq i \leq n)$ and $k = 1 + \sum_{1 \leq i \leq n} k_i$.

The string language $L(G)$ generated by a hg, $G = (V_N, V_T, S, P)$ is defined as

$$L(G) = \left\{ uv \in V_T^* \;\middle|\; S \underset{G}{\Longrightarrow} u_{\uparrow}v \right\}$$

where $\underset{G}{\Longrightarrow} = \bigcup_{k \geq 0} \overset{k}{\underset{G}{\Longrightarrow}}$.

Example 2.1 The hg $G = (\{ S, T \}, \{ a, b, c, d \}, S, P)$ generates the language $\{ a^n b^n c^n d^n \mid n \geq 0 \}$ where P is as follows. Note that ϵ denotes the empty string.

$$P = \left\{ \; S \rightarrow C_{1,1}(\epsilon_{\uparrow}\epsilon), \qquad S \rightarrow C_{2,3}(a_{\uparrow}\epsilon, T, d_{\uparrow}\epsilon), \qquad T \rightarrow W(S, b_{\uparrow}c) \; \right\}$$

Consider the following derivation of the string $aabbccdd$.

$$\begin{array}{rrcl} & S & \overset{1}{\underset{G}{\Longrightarrow}} & C_{1,1}(\epsilon_{\uparrow}\epsilon) = \epsilon_{\uparrow}\epsilon \\ \text{hence} & T & \overset{2}{\underset{G}{\Longrightarrow}} & W(\epsilon_{\uparrow}\epsilon, b_{\uparrow}c) = b_{\uparrow}c \\ \text{hence} & S & \overset{3}{\underset{G}{\Longrightarrow}} & C_{2,3}(a_{\uparrow}\epsilon, b_{\uparrow}c, d_{\uparrow}\epsilon) = ab_{\uparrow}cd \\ \text{hence} & T & \overset{4}{\underset{G}{\Longrightarrow}} & W(ab_{\uparrow}cd, b_{\uparrow}c) = abb_{\uparrow}ccd \\ \text{hence} & S & \overset{5}{\underset{G}{\Longrightarrow}} & C_{2,3}(a_{\uparrow}\epsilon, abb_{\uparrow}ccd, d_{\uparrow}\epsilon) = aabb_{\uparrow}ccdd \end{array}$$

2.2 Tree Adjoining Grammars

Tree adjoining grammars (tag) were introduced in [8] and their formal properties investigated in [7, 22]. The linguistic use of tag is discussed in [12, 13, 10, 11, 18, 14].

Let $\mathcal{N}_+$ denote the set of positive integers. D is a tree domain if it is a nonempty finite subset of $\mathcal{N}_+^*$ such that if $d \in D$ and $d = d_1 d_2$ then $d_1 \in D$ and if $di \in D$ where $i \in \mathcal{N}_+$ then $dj \in D$ for all $1 \leq j \leq i$. A tree γ is denoted by a partial function $\gamma : \mathcal{N}_+^* \rightarrow \Sigma$ where $dom\,(\gamma)$ is a tree domain and Σ is the set of node labels (and $dom\,(\gamma)$ is the, usual, tree domain γ). We say that the elements of a tree domain are addresses of the nodes of the tree and $\gamma(d)$ is the label of the node with address d. Note that ϵ is the address of the root node of the tree. If $d \in dom\,(\gamma)$ for some tree γ then γ/d, the subtree rooted at d in γ is defined such that for all $d' \in \mathcal{N}_+^*$ $\gamma/d(d') = \gamma(dd')$.

Tree adjoining grammars manipulate trees whose nodes are labelled either by terminal symbols or by triples of the form $\langle A, \mathsf{sa}, \mathsf{oa} \rangle$ where A is a nonterminal symbol, sa is a set of tree labels (that determines which of the trees of the grammar can be adjoined at that node) and oa is either true (indicating that adjunction is obligatory) or false (indicating that adjunction is optional). We call sa and oa the adjunction constraints at that node. A node at which the value of oa is true is said to have an OA constraint. A node at which the value of $\mathsf{sa} = \emptyset$ is said to have an NA constraint.

Let V_N be a set of nonterminal symbols, V_T a set of terminal symbols, V_L a set of tree labels and $V_T^\epsilon = V_T \cup \{ \epsilon \}$

Initial Trees For each $A \in V_N$, $init\,(V_N, V_T, V_L, A)$ is the set of trees $\alpha : D_\alpha \rightarrow V_T^\epsilon \cup (V_N \times 2^{V_L} \times \{ \mathsf{true}, \mathsf{false} \})$ where D_α is a tree domain and the following hold.

- The root of α is labelled $\langle A, \mathsf{sa}, \mathsf{oa} \rangle$ for some $\mathsf{sa} \subseteq V_L$ and $\mathsf{oa} \in \{ \mathsf{true}, \mathsf{false} \}$.
- All internal nodes of α are labelled $\langle B, \mathsf{sa}, \mathsf{oa} \rangle$ for some $B \in V_N$, $\mathsf{sa} \subseteq V_L$ and $\mathsf{oa} \in \{ \mathsf{true}, \mathsf{false} \}$.
- All leaf nodes of α are labelled by some $u \in V_T^\epsilon$.

Auxiliary Trees For each $A \in V_N$, $aux\,(V_N, V_T, V_L, A)$ is the set of trees $\beta : D_\beta \rightarrow V_T^\epsilon \cup (V_N \times 2^{V_L} \times \{ \mathsf{true}, \mathsf{false} \})$ where D_β is a tree domain and the following hold.

- The root of β is labelled $\langle A, \mathsf{sa}, \mathsf{oa}\rangle$ for some $\mathsf{sa} \subseteq V_L$ and $\mathsf{oa} \in \{\mathsf{true}, \mathsf{false}\}$.
- All internal nodes of β are labelled $\langle B, \mathsf{sa}, \mathsf{oa}\rangle$ for some $B \in V_N$, $\mathsf{sa} \subseteq V_L$ and $\mathsf{oa} \in \{\mathsf{true}, \mathsf{false}\}$.
- All leaf nodes of β except one are labelled by some $u \in V_T^\epsilon$. The remaining leaf node is called the foot node and is labelled $\langle A, \mathsf{sa}, \mathsf{oa}\rangle$ for some $\mathsf{sa} \subseteq V_L$ and $\mathsf{oa} \in \{\mathsf{true}, \mathsf{false}\}$. The address of the foot node of β is denoted $ft(\beta)$.

Let $aux(V_N, V_T, V_L) = \bigcup_{A \in V_N} aux(V_N, V_T, V_L, A)$.

Elementary Trees $elem(V_N, V_T, V_L, A) = init(V_N, V_T, V_L, A) \cup aux(V_N, V_T, V_L)$.

For each $\beta \in aux(V_N, V_T, V_L)$ let $spine(\beta) = \left\{ d \;\middle|\; ft(\beta) = dd' \text{ for some } d' \in \mathcal{N}_+^* \right\}$, i.e., $spine(\beta)$ are the addresses on the **spine** of β which is the path from the root to the foot node of β.

We now define the **tree adjunction** operation

$$\nabla\colon elem(V_N, V_T, V_L, A) \times aux(V_N, V_T, V_L) \times \mathcal{N}_+^* \to elem(V_N, V_T, V_L, A)$$

for every $A \in V_N$. For $\gamma \in elem(V_N, V_T, V_L, A)$, $\beta \in aux(V_N, V_T, V_L)$, $d \in dom(\gamma)$ we have $\gamma' = \nabla(\gamma, \beta, d)$ where for all $d' \in \mathcal{N}_+^*$

$$\gamma'(d') = \begin{cases} \gamma(d') & \text{if } d \text{ is not a prefix of } d' \\ \beta(d'') & \text{if } d' = dd'' \text{ and } d'' \in dom(\beta) \\ \gamma(dd'') & \text{if } d' = dd_f d'' \text{ such that } d'' \neq \epsilon \text{ and } d_f = ft(\beta) \end{cases}$$

The adjunction operation is shown in Figure 1.

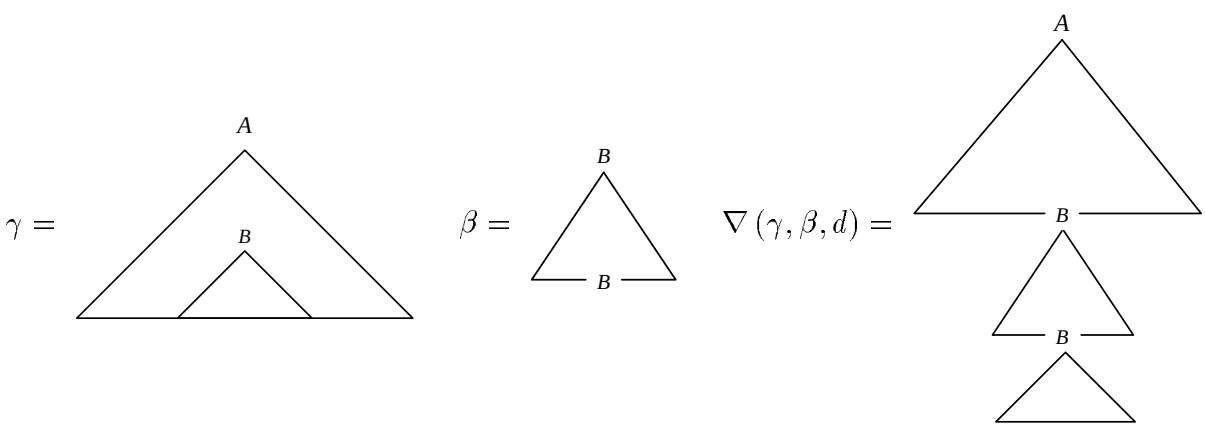

Figure 1: Adjunction

Definition 2.2 A tag is a seven tuple $G = (V_N, V_T, V_L, S, \mathcal{I}, \mathcal{A}, \text{-})$ where
V_N is a finite set of nonterminals,
V_T is a finite set of terminals,
V_L is a finite set of tree labels,
$S \in V_N$ is a distinguished nonterminal,
$\mathcal{I}$ is a finite subset of $init(V_N, V_T, V_L, S)$,
$\mathcal{A}$ is a finite subset of $aux(V_N, V_T, V_L)$ and
$\text{-}\colon \mathcal{A} \to V_L$ is a bijection, the auxiliary tree labelling function.

We use $\overline{\beta}$ to denote the result of applying $_$ to β.

Given a tag $G = (V_N, V_T, V_L, S, \mathcal{I}, \mathcal{A}, _)$ $\underset{G}{\Longrightarrow}$ is defined as follows.

If $\gamma \in elem\,(V_N, V_T, V_L, A)$, $\gamma(d) = \langle B, \mathsf{sa}, \mathsf{oa}\rangle$, $\beta \in \mathcal{A} \cap aux\,(V_N, V_T, V_L, B)$, and $\overline{\beta} \in \mathsf{sa}$ then

$$\gamma \underset{G}{\Longrightarrow} \nabla\,(\gamma, \beta, d)$$

Note that the same nonterminal, B, must label the adjunction node and the root and foot of the adjoined tree. Note that for all $\gamma \in elem\,(V_N, V_T, V_L, A)$, $\beta_1, \beta_2 \in aux\,(V_N, V_T, V_L)$, $d_1 \in dom\,(\gamma)$, and $d_2 \in dom\,(\beta_1)$

$$\nabla\,(\nabla\,(\gamma, \beta_1, d_1)\,, \beta_2, d_1 d_2) = \nabla\,(\gamma, \nabla\,(\beta_1, \beta_2, d_2)\,, d_1)$$

Thus, for all $\gamma \in elem\,(V_N, V_T, V_L, A)$, $\beta_1, \beta_2 \in aux\,(V_N, V_T, V_L)$ and $d \in dom\,(\gamma)$ if $\gamma \underset{G}{\Longrightarrow} \nabla\,(\gamma, \beta_1, d)$ and $\beta_1 \overset{*}{\underset{G}{\Longrightarrow}} \beta_2$ then $\gamma \overset{*}{\underset{G}{\Longrightarrow}} \nabla\,(\gamma, \beta_2, d)$ where $\overset{*}{\underset{G}{\Longrightarrow}}$ is the reflexive, transitive closure of $\underset{G}{\Longrightarrow}$.

A tree γ has no OA nodes if for all $d \in dom\,(\gamma)$ either $\gamma(d) = \langle A, \mathsf{sa}, \mathsf{false}\rangle$ for some A and sa or $\gamma(d) = u$ for some $u \in V_T^{\epsilon}$.

The tree language $T(G)$ generated by $G = (V_N, V_T, V_L, S, \mathcal{I}, \mathcal{A}, _)$ is defined as

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

In defining $yield\,(\gamma)$, the yield of a tree γ, we consider only symbols in V_T and V_N, i.e., we do not give the SA or OA constraints of nodes involving nonterminals. For a tree γ

- if γ consists of a single node then if $\gamma(\epsilon) = u$ for some $u \in V_T^{\epsilon}$ then the $yield\,(\gamma) = u$ and if $\gamma(\epsilon) = \langle A, \mathsf{sa}, \mathsf{oa}\rangle$ for some A, sa and oa then the $yield\,(\gamma) = A$.
- Otherwise if the root of γ has k children ($k > 0$) then

$$yield\,(\gamma) = yield\,(\gamma/1) \ldots yield\,(\gamma/k)$$

The string language, $L(G)$, generated by G, is defined as

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

When displaying a tree graphically we use several conventions. The nonterminal or terminal component of a node label is always shown. The adjunction constraints for a node labelled $\langle A, \mathsf{sa}, \mathsf{oa}\rangle$ are specified as follows. When sa is the empty set the node is annotated NA. The case in which sa includes the labels of *all* auxiliary trees in the grammar that are in $aux\,(V_N, V_T, V_L, A)$ (note that A is the same) is the default so no annotation is used. Otherwise, the set sa is given in full. The case in which oa is false is the default and no annotation is used. When oa is true the node is annotated OA.

Example 2.2 The tag

$$G = (\{\, S \,\}, \{\, a, b, c, d \,\}, \left\{ \overline{\beta} \right\}, S, \{\, \alpha \,\}, \{\, \beta \,\}, _)$$

generates the language $\{\, a^n b^n c^n d^n \mid n \geq 0 \,\}$. The trees α and β and a derivation of the string $aabbccdd$ are given in Figure 2.

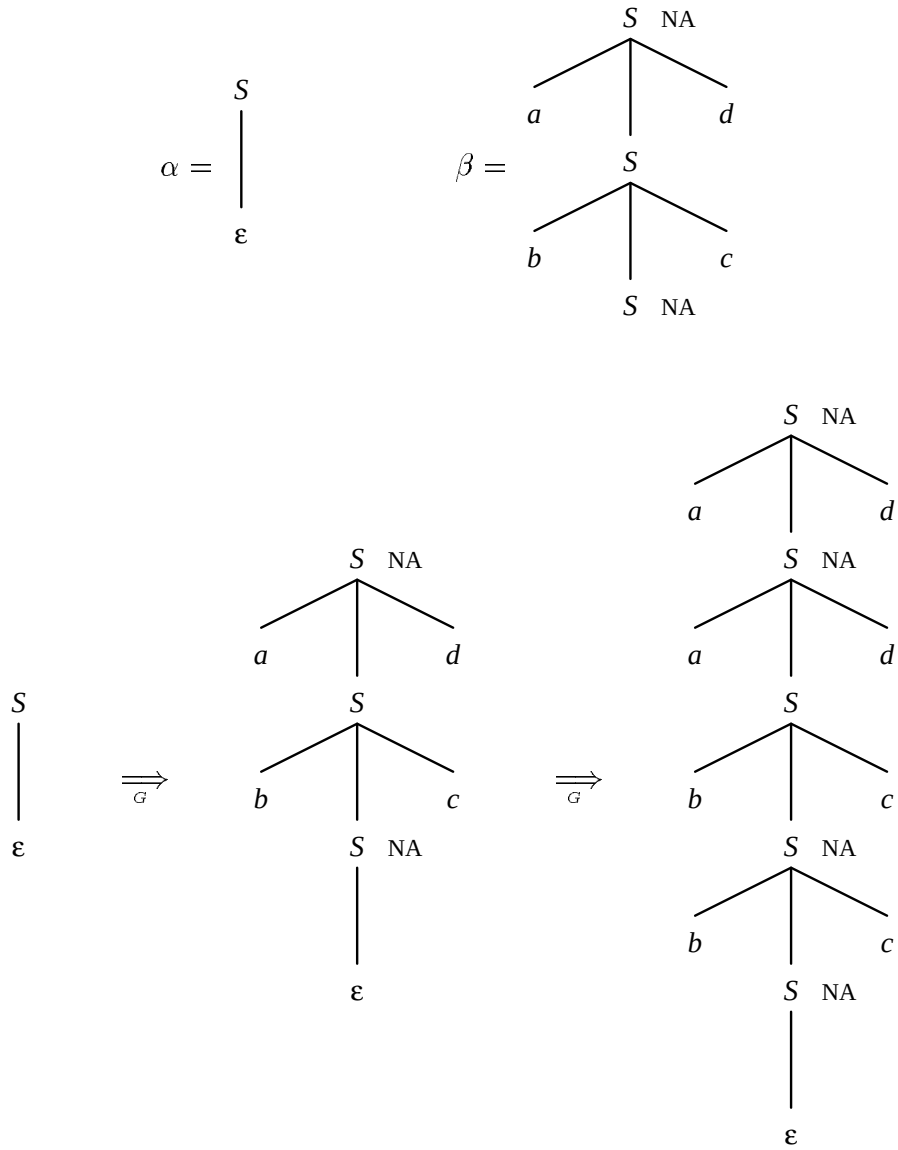

Figure 2: Example of a tag

2.3 Linear Indexed Grammars

Linear indexed grammars (lig) where first discussed, though not named, in [5][2]. In a lig strings are derived from nonterminals with an associated stack denoted $A[l_1 \ldots l_n]$ where A is a nonterminal, each l_i is a stack symbol for $1 \leq i \leq n$, and l_n is the top of the stack. $A[]$ denotes the nonterminal A associated with the empty stack. Since, during a derivation, stacks can grow to be of unbounded size we need some way of partially specifying unbounded stacks in lig productions. We use $A[\circ\circ\, l_1 l_2 \ldots l_n]$ to denote the nonterminal A associated with any stack η those top n symbols are $l_1, l_2, \ldots, l_n$ where $n \geq 0$. We denote the set of all nonterminals in V_N associated with stacks whose symbols come from V_I by the expression $V_N[V_I^*]$.

Definition 2.3 A lig is a five tuple $G = (V_N, V_T, V_I, S, P)$ where
V_N is a finite set of nonterminals,

[2]In [4] the name linear indexed grammars was also used to name a different restriction of indexed grammars [1] in which only one nonterminal can appear on the right of production.

V_T is a finite set of terminals,
V_I is a finite set of indices (stack symbols),
$S \in V_N$ is the start symbol and
P is a finite set of productions, having one of the following two forms.

$$A[\circ\circ\,\eta] \rightarrow \psi A'[\circ\circ\,\eta']\psi' \qquad A[\eta] \rightarrow \psi$$

where $A, A' \in V_N$, $\eta, \eta' \in V_I^*$, $\psi, \psi' \in (V_N[V_I^*] \cup V_T)^*$.

Given a lig $G = (V_N, V_T, V_I, S, P)$ $\underset{G}{\Longrightarrow}$ is defined as follows.

- If $A[\circ\circ\,\eta] \rightarrow \psi A'[\circ\circ\,\eta']\psi' \in P$ then for all $\psi_1, \psi_2 \in (V_N[V_I^*] \cup V_T)^*$ and $\eta'' \in V_I^*$ we have

  $$\psi_1 A[\eta''\eta]\psi_2 \underset{G}{\Longrightarrow} \psi_1\psi A'[\eta''\eta']\psi'\psi_2$$

  In this derivation step we say that the occurrence of $A'[\eta''\eta']$ shown on the right is the distinguished successor of the occurrence of $A[\eta''\eta]$ shown on the left.

- If $A[\eta] \rightarrow \psi \in P$ then for all $\psi_1, \psi_2 \in (V_N[V_I^*] \cup V_T)^*$

  $$\psi_1 A[\eta]\psi_2 \underset{G}{\Longrightarrow} \psi_1\psi\psi_2$$

Let distinguished descendent be the reflexive, transitive closure of distinguished successor.

The language, $L(G)$, generated by G is

$$\left\{ w \in V_T^* \;\middle|\; S[] \overset{*}{\underset{G}{\Longrightarrow}} w \right\}$$

where $\overset{*}{\underset{G}{\Longrightarrow}}$ is the reflexive, transitive closure of $\underset{G}{\Longrightarrow}$.

Example 2.3 The lig, $G = (\{S, T\}, \{a, b, c, d\}, \{l\}, S, P)$, generates the language $\{a^n b^n c^n d^n \mid n \geq 0\}$ where P is as follows.

$$P = \left\{ \begin{array}{ll} S[\circ\circ] \rightarrow aS[\circ\circ\,l]d, & S[\circ\circ] \rightarrow T[\circ\circ], \\ T[\circ\circ\,l] \rightarrow bT[\circ\circ]c, & T[] \rightarrow \epsilon \end{array} \right\}$$

The following is a derivation of the string $aabbccdd$.

$$\begin{aligned} S[] &\underset{G}{\Longrightarrow} aS[l]d \\ &\underset{G}{\Longrightarrow} aaS[ll]dd \\ &\underset{G}{\Longrightarrow} aaT[ll]dd \\ &\underset{G}{\Longrightarrow} aabT[l]cdd \\ &\underset{G}{\Longrightarrow} aabbT[]ccdd \\ &\underset{G}{\Longrightarrow} aabbccdd \end{aligned}$$

2.4 Combinatory Categorial Grammars

Combinatory categorial grammars (ccg) are an extension of classical categorial grammars [2] in which function composition is used in addition to application. The version of ccg that we consider was developed by Steedman [21, 20].

The objects that derive terminals in a ccg are categories. The set of categories, $cat\,(V_N)$, over the alphabet V_N, is the smallest set such that:

- $V_N \subseteq cat\,(V_N)$ (members of V_N are the atomic categories) and
- if c_1 and c_2 are categories in $cat\,(V_N)$ then (c_1/c_2) and $(c_1\backslash c_2)$ are in $cat\,(V_N)$.

  Intuitively, values having the categories (c_1/c_2) and $c_1\backslash c_2$ are functions that can combine with a value of category c_2 at their right and left, respectively, to give a value having the category c_1.

For each category $c \in cat\,(V_N)$ we define its target category, $target\,(c) \in V_N$, argument categories $args\,(c) \subseteq cat\,(V_N)$ and arity $arity\,(c) \in \mathcal{N}$.

- For $c \in V_N$ $target\,(c) = c$, $args\,(c) = \phi$ and $arity\,(c) = 0$.
- For categories (c_1/c_2) and $(c_1\backslash c_2)$
  - $target\,((c_1/c_2)) = target\,((c_1\backslash c_2)) = target\,(c_1)$,
  - $args\,((c_1/c_2)) = args\,((c_1\backslash c_2)) = args\,(c_1) \cup \{\, c_2 \,\}$ and
  - $arity\,((c_1/c_2)) = arity\,((c_1\backslash c_2)) = 1 + arity\,(c_1)$.

For example,

$$\begin{array}{l}
target\,((((A/B)\backslash(C/D))\backslash E)) = A \\
args\,((((A/B)\backslash(C/D))\backslash E)) = \{\, E, (C/D), B \,\} \\
arity\,((((A/B)\backslash(C/D))\backslash E)) = 3 \\
target\,((((S\backslash NP)/NP)/PP)) = S \\
args\,((((S\backslash NP)/NP)/PP)) = \{\, PP, NP \,\} \\
arity\,((((S\backslash NP)/NP)/PP)) = 3
\end{array}$$

An object of category c can combined with $arity\,(c)$ objects (whose categories are in $args\,(c)$) to give an object having the atomic category $target\,(c)$.

Since the size of categories derived in ccg can be unbounded we need some way of denoting unbounded categories. We use terms of the form $(\ldots(x|_1x_1)|_2x_2)\ldots x_{n-1})|_nx_n)$ where $n \geq 0$, $x, x_1, \ldots, x_n$ are variables ranging over $cat\,(V_N)$ and $|_i \in \{\, /, \backslash \,\}$ $(1 \leq i \leq n)$. In addition, for each variable x and atomic category $A \in V_N$ we have the target-restricted variable $\overset{A}{x}$ which ranges over $\{\, c \in cat\,(V_N) \mid target\,(c) = A \,\}$ and we say that the variable x has been target restricted to A.

We define the set of **combinatory schemata** $\mathcal{C}$ as follows. $\mathcal{C} = \bigcup_{n\geq 0}(F_n \cup B_n)$ where for each $n \geq 0$

- the set of forward schemata F_n consists of all

$$(x/y)\ (\ldots(y|_1z_1)|_2\ldots|_nz_n) \rightarrow (\ldots(x|_1z_1)|_2\ldots|_nz_n)$$

  with $|_i \in \{\, /, \backslash \,\}$

- the set of backward schemata B_n consists of all

$$(\ldots(y|_1 z_1)|_2 \ldots |_n z_n)\ (x\backslash y) \rightarrow (\ldots(x|_1 z_1)|_2 \ldots |_n z_n)$$

  with $|_i \in \{\,/, \backslash\,\}$.

In these schemata $(x|y)$ and $(\ldots(y|_1 z_1)|_2 \ldots |_n z_n)$ are called the primary and secondary components, respectively.

Each grammar restricts itself to the use of a finite selection of *instances* of schemata in $\mathcal{C}$ which we call combinatory rules. For each $r \in \mathcal{C}$ such that $(x|y)$ and $(\ldots(y|_1 z_1)|_2 \ldots |_n z_n)$ are the primary and secondary component of r, respectively let

$$\begin{aligned} r(V_N) = \{r[(x,w),(y,w_0),(z_1,w_1),\ldots,(z_n,w_n)] \mid \quad & w \in \{\,x\,\} \cup \left\{\, \overset{A}{x} \mid A \in V_N \,\right\} \\ & w_0 \in \{\,y\,\} \cup cat\,(V_N) \text{ and} \\ & w_i \in \{\,z_i\,\} \cup cat\,(V_N) \text{ for } 1 \leq i \leq n\} \end{aligned}$$

where $r[(x_1,w_1),\ldots,(x_k,w_k)]$ denotes the result of substituting w_i for x_i in r $(1 \leq i \leq k)$.

The set of **combinatory rules** $\mathcal{C}(V_N)$ is as follows.

$$\mathcal{C}(V_N) = \bigcup_{r \in \mathcal{C}} r(V_N)$$

We extend the terms primary and secondary components to members of $\mathcal{C}(V_N)$ in the obvious way.

$c'c'' \rightarrow c'''$ is a **ground instance** of $r \in \mathcal{C}(V_N)$ if there are $c, c_0, \ldots, c_n \in cat\,(V_N)$ such that $(c'c'' \rightarrow c''') = r[(w,c)(w_0,c_0),(w_1,c_1),\ldots,(w_n,c_n)]$ and the following hold.

- The primary and secondary components of r are $(w|w_0)$ and $(\ldots(w_0|_1 w_1)|_2 \ldots |_n w_n)$, respectively.
- If w is a target restricted variable $\overset{A}{x}$ then $target\,(c) = A$.
- If $w_i \in cat\,(V_N)$ then $c_i = w_i$ $(0 \leq i \leq n)$.

Definition 2.4 A ccg is a five tuple (V_N, V_T, S, f, R) where
V_N is a finite set of nonterminals (atomic categories),
V_T is a finite set of terminals (lexical items),
S is a distinguished member of V_N,
f is a function that maps elements of V_T^ϵ to finite subsets of $cat\,(V_N)$ and
R is a finite subset of $\mathcal{C}(V_N)$.

Each step in the derivations of a ccg, $G = (V_N, V_T, S, f, R)$, involves the use of a combinatory rule in R. For every $\varphi_1, \varphi_2 \in cat\,(V_N)^*$

$$\varphi_1 c \varphi_2 \underset{G}{\Longrightarrow} \varphi_1 c' c'' \varphi_2$$

if there is some $r \in R$ such that $c'c'' \rightarrow c$ is a ground instance of r. Whichever of c' or c'' is the primary component of the rule is said to be the primary child of c with respect to this derivation. Let the primary descendent relation be the reflexive, transitive closure of the primary child relation. Note that in defining the derivations we are applying the rules backwards.

The string language, $L(G)$, generated by G is defined as follows.

$$L(G) = \{w_1 \ldots w_n | \quad S \overset{*}{\underset{G}{\Longrightarrow}} c_1 \ldots c_n \text{ for some } c_1, \ldots, c_n \in cat\,(V_N) \\ \text{and } c_i \in f(w_i) \text{ where } w_i \in V_T^{\epsilon}\ (1 \leq i \leq n)\}$$

Example 2.4 The ccg, $G = (\{\, S, T, A, B, D \,\}, \{\, a, b, c, d \,\}, S, f, R)$, generates the language $\{\, a^n b^n c^n d^n \mid n \geq 0 \,\}$ where f and R are as follows. Note that in this example parentheses have been omitted except when needed to override the left associativity of the two slashes.

$$f(a) = \{\, (A/D) \,\} \qquad f(b) = \{\, B \,\} \qquad f(d) = \{\, D \,\}$$

$$f(c) = \{\, (T\backslash A/T\backslash B) \,\} \qquad f(\epsilon) = \{\, (S/T), T \,\}$$

$$R = \left\{ \begin{array}{rcll} (\overset{S}{x}/T)\ (T\backslash A/T\backslash B) & \rightarrow & (\overset{S}{x}\backslash A/T\backslash B) & \text{(rule 1)} \\ (A/D)\ (\overset{S}{x}\backslash A) & \rightarrow & (\overset{S}{x}/D) & \text{(rule 2)} \\ (\overset{S}{x}/y)\ y & \rightarrow & \overset{S}{x} & \text{(rule 3)} \\ y\ (\overset{S}{x}\backslash y) & \rightarrow & \overset{S}{x} & \text{(rule 4)} \end{array} \right\}$$

The following is a derivation of a string of categories that map to the string *aabbccdd* using the definition of $L(G)$ above.

$$\begin{array}{lll} S & \underset{G}{\Longrightarrow} (S/D)\ D & \text{rule 3} \\ & \underset{G}{\Longrightarrow} (A/D)\ (S\backslash A)\ D & \text{rule 2} \\ & \underset{G}{\Longrightarrow} (A/D)\ (S\backslash A/D)\ D\ D & \text{rule 3} \\ & \underset{G}{\Longrightarrow} (A/D)\ (A/D)\ (S\backslash A\backslash A)\ D\ D & \text{rule 2} \\ & \underset{G}{\Longrightarrow} (A/D)\ (A/D)\ (S\backslash A\backslash A/T)\ T\ D\ D & \text{rule 3} \\ & \underset{G}{\Longrightarrow} (A/D)\ (A/D)\ B\ (S\backslash A\backslash A/T\backslash B)\ T\ D\ D & \text{rule 4} \\ & \underset{G}{\Longrightarrow} (A/D)\ (A/D)\ B\ (S\backslash A/T)\ (T\backslash A/T\backslash B)\ T\ D\ D & \text{rule 1} \\ & \underset{G}{\Longrightarrow} (A/D)\ (A/D)\ B\ B\ (S\backslash A/T\backslash B)\ (T\backslash A/T\backslash B)\ T\ D\ D & \text{rule 4} \\ & \underset{G}{\Longrightarrow} (A/D)\ (A/D)\ B\ B\ (S/T)\ (T\backslash A/T\backslash B)\ (T\backslash A/T\backslash B)\ T\ D\ D & \text{rule 1} \end{array}$$

Example 2.5 Figure 3 gives a derivation for the sentence *John might eat apples*. Note that in the tree parentheses have only been included when needed to override the left associativity of the slashes. The rule B_0 is applied with $(S\backslash NP)$ and NP as the primary and secondary constituents, respectively. The rule F_0 is applied with $((S\backslash NP)/NP)$ and NP as the primary and secondary constituents, respectively. The rule F_1 is applied with $((S\backslash NP)/(S\backslash NP))$ and $((S\backslash NP)/NP)$ as the primary and secondary constituents, respectively. *John* and *apples* are in $f(NP)$, *eat* is in $f(((S\backslash NP)/NP))$ and *might* is in $f(((S\backslash NP)/(S\backslash NP)))$.

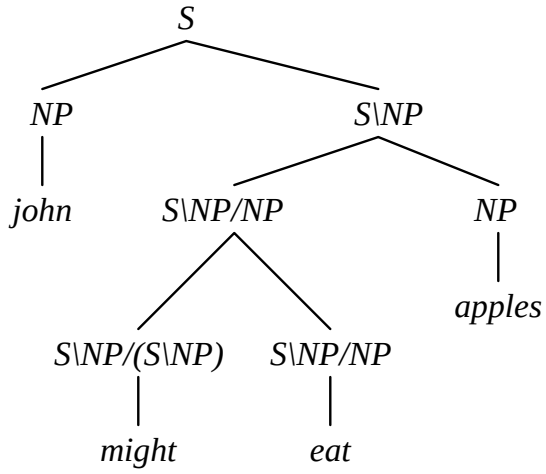

Figure 3: Example ccg derivation

3 Proofs of Equivalence

Let the classes of languages generated by ccg, hg, lig, and tag be ccl, hl, lil, and tal, respectively. In this section we show that they are identical by proving that ccl $\subseteq$ lil, lil $\subseteq$ hl, hl $\subseteq$ tal and tal $\subseteq$ ccl.

3.1 ccl $\subseteq$ lil

We show that only one of the variables used in the combinatory rules of a ccg ranges over an unbounded number of categories. The use of the other variables can be viewed as merely a convenient way of describing a finite number of alternative rules.

Lemma 3.1 Given a ccg, $G = (V_N, V_T, S, f, R)$, $c, c_1, \ldots, c_n \in cat\,(V_N)$, and $w_1, \ldots, w_n \in V_T^\epsilon$, if $c \overset{*}{\underset{G}{\Longrightarrow}} c_1 \ldots c_n$ and $c_i \in f(w_i)$ $(1 \leq i \leq n)$ then $args\,(c) \subseteq args\,(G)$ where

$$args\,(G) = \{\, c \mid c \in args\,(c') \text{ for some } c' \in f(w) \text{ and } w \in V_T^\epsilon \,\}$$

Proof This follows by trivial induction on the length of the derivation in G. ∎

Lemma 3.2 For every ccg, $G = (V_N, V_T, S, f, R)$, there is an equivalent ccg $G' = (V_N, V_T, S, f, R')$ such that the rules in R' have one of the following two forms.

$$(\overset{A}{x}/c)\,(\ldots(c|_1c_1)|_2\ldots|_nc_n) \rightarrow (\ldots(\overset{A}{x}|_1c_1)|_2\ldots|_nc_n)$$

$$(\ldots(c|_1c_1)|_2\ldots|_nc_n)\,(\overset{A}{x}\backslash c) \rightarrow (\ldots(\overset{A}{x}|_1c_1)|_2\ldots|_nc_n)$$

where $n \geq 0$, $|_i \in \{\,\backslash, /\,\}$ $(1 \leq i \leq n)$, $A \in V_N$ and $c, c_1, \ldots, c_n \in cat\,(V_N)$.

Proof From a ccg $G = (V_N, V_T, S, f, R)$ we construct the ccg $G' = (V_N, V_T, S, f, R')$ satisfying the conditions of Lemma 3.2 such that $L(G) = L(G')$.

For each $r \in R$ from Lemma 3.1 we know that for any ground instance of the rule r used to derive a string in $L(G)$ the categories substituted for variables in the secondary component of r

must be members of the finite set $\mathit{args}\,(G)$. Thus, for every $A \in V_N$ and $c_0, \ldots, c_n \in \mathit{args}\,(G)$ include the rule $r\left[(w, \overset{A}{x}), (w_0, c_0), (w_1, c_1), \ldots, (w_n, c_n)\right]$ in R' when the following conditions hold.

- The primary and secondary components of r are $(w|w_0)$ and $(\ldots(w_0|_1 w_1)|_2 \ldots |_n w_n)$, respectively.
- w is either the variable x or the target restricted variable $\overset{A}{x}$.
- If $w_i \in \mathit{cat}\,(V_N)$ then $w_i = c_i$ $(0 \leq i \leq n)$.

∎

Let $G = (V_N, V_T, S, f, R)$ be a ccg satisfying the conditions of Lemma 3.2. We construct an equivalent lig, $G' = (V_N, V_T, V_I, S, P)$ where $V_I = \{\,/, \backslash, (,)\,\} \cup V_N$. The proof is straightforward since the rules in R can be seen as simple notational variants of lig productions. For $c \in \mathit{cat}\,(V_N)$ and $| \in \{\,\backslash, /\,\}$ we call $|c$ a directional category. Consider the category $(\ldots(A|_1 c_1)\ldots|_n c_n)$ where $A \in V_N$ is the target category and $c_1, \ldots, c_n$ are the arguments of the category. This can be viewed as the atomic category A associated with n arguments $|_i c_i$ that are directional categories $(1 \leq i \leq n)$. The combinatory rules of G manipulate the string of directional categories in a stack-like way that can be imitated by the lig G'.

We begin by defining a translation function $\mathit{enc} : \mathit{cat}\,(V_N) \rightarrow V_N[V_I^*]$ that determines how the categories of G will be encoded in the lig G'.

- For $A \in V_N$ let $\mathit{enc}\,(A) = A[]$.
- $\mathit{enc}\,((c_1/c_2)) = A[\eta/c_2]$ and $\mathit{enc}\,((c_1\backslash c_2)) = A[\eta\backslash c_2]$ where $\mathit{enc}\,(c_1) = A[\eta]$.

For example, $\mathit{enc}\,(((A/(B\backslash C))\backslash D)) = A[/(B\backslash C)\backslash D]$.

- Consider the forward rule

  $$(\overset{A}{x}/c_0)\,(\ldots(c_0|_1 c_1)|_2 \ldots |_n c_n) \rightarrow (\ldots(\overset{A}{x}|_1 c_1)|_2 \ldots |_n c_n) \in R$$

  where $n \geq 0$ and for $1 \leq i \leq n$, $|_i \in \{\,\backslash, /\,\}$, $A \in V_N$ and $c_0, c_1, \ldots, c_n \in \mathit{cat}\,(V_N)$. Corresponding to this rule we add the following production to P.

  $$A[\circ\circ\,\eta] \rightarrow A[\circ\circ\,/c_0]\,B[\eta']$$

  where $\eta = |_1 c_1 \ldots |_n c_n$ and $\mathit{enc}\,((\ldots(c_0|_1 c_1)|_2 \ldots |_n c_n)) = B[\eta']$
- The corresponding backward rule is treated in a similar way.
- If $c \in f(w)$ for $w \in V_T^\epsilon$ then let $A[\eta] \rightarrow w \in P$ where $\mathit{enc}\,(c) = A[\eta]$.

Lemma 3.3 $\quad c \underset{G}{\Longrightarrow} c_1\, c_2$ if and only if $\mathit{enc}\,(c) \underset{G'}{\Longrightarrow} \mathit{enc}\,(c_1)\,\mathit{enc}\,(c_2)$

Proof This follows trivially from the above construction. ∎

Thus, $S \overset{*}{\underset{G}{\Longrightarrow}} c_1 \ldots c_n$ if and only if $S[] \overset{*}{\underset{G'}{\Longrightarrow}} \mathit{enc}\,(c_1) \ldots \mathit{enc}\,(c_n)$ and for each $1 \leq i \leq n$ $\mathit{enc}\,(c_i) \rightarrow w$ if and only if $c_i \in f(w)$. Thus it is clear that $L(G) = L(G')$.

Example 3.1 $G' = (V_N, V_T, S, f, R')$ is equivalent to $G = (V_N, V_T, S, f, R)$ of Example 2.4 satisfying the conditions of Lemma 3.2. The above construction would convert G' into the lig

$$G'' = (V_N, V_T, \{\, /, \backslash, (,) \,\} \cup V_N, S, P)$$

Note that $\mathit{args}\,(G) = \{\, A, B, D, T \,\}$.

First, we show how the rules in R' are derived from those in R.

Rules in R	Corresponding Rule(s) in R'
$(\overset{s}{x}/T)\ (T\backslash A/T\backslash B) \rightarrow (\overset{s}{x}\backslash A/T\backslash B)$	unchanged
$(A/D)\ (\overset{s}{x}\backslash A) \rightarrow (\overset{s}{x}/D)$	unchanged
$(\overset{s}{x}/y)\ y \rightarrow \overset{s}{x}$	$\begin{cases} (\overset{s}{x}/A)\ A \rightarrow \overset{s}{x} \\ (\overset{s}{x}/B)\ B \rightarrow \overset{s}{x} \\ (\overset{s}{x}/D)\ D \rightarrow \overset{s}{x} \\ (\overset{s}{x}/T)\ T \rightarrow \overset{s}{x} \end{cases}$
$y\ (\overset{s}{x}\backslash y) \rightarrow \overset{s}{x}$	$\begin{cases} A\ (\overset{s}{x}\backslash A) \rightarrow \overset{s}{x} \\ B\ (\overset{s}{x}\backslash B) \rightarrow \overset{s}{x} \\ D\ (\overset{s}{x}\backslash D) \rightarrow \overset{s}{x} \\ T\ (\overset{s}{x}\backslash T) \rightarrow \overset{s}{x} \end{cases}$

Second, we show those productions in P that are derived from f.

Components of f	Corresponding productions in P
$(A/D) \in f(a)$	$A[/D] \rightarrow a$
$B \in f(b)$	$B[] \rightarrow b$
$D \in f(d)$	$D[] \rightarrow d$
$(T\backslash A/T\backslash B) \in f(c)$	$T[\backslash A/T\backslash B] \rightarrow c$
$(S/T) \in f(\epsilon)$	$S[/T] \rightarrow \epsilon$
$T \in f(\epsilon)$	$T[] \rightarrow \epsilon$

Third, we show how the remaining productions in P are derived from the rules in R'.

Rules in R'	Corresponding Rules in P
$(\overset{s}{x}/T)\ (T\backslash A/T\backslash B) \rightarrow (\overset{s}{x}\backslash A/T\backslash B)$	$S[\circ\circ\backslash A/T\backslash B] \rightarrow S[\circ\circ/T]\ T[\backslash A/T\backslash B]$
$(A/D)\ (\overset{s}{x}\backslash A) \rightarrow (\overset{s}{x}/D)$	$S[\circ\circ/D] \rightarrow A[/D]\ S[\circ\circ\backslash A]$
$(\overset{s}{x}/A)\ A \rightarrow \overset{s}{x}$	$S[\circ\circ] \rightarrow S[\circ\circ/A]\ A[]$
$(\overset{s}{x}/B)\ B \rightarrow \overset{s}{x}$	$S[\circ\circ] \rightarrow S[\circ\circ/B]\ B[]$
$(\overset{s}{x}/D)\ D \rightarrow \overset{s}{x}$	$S[\circ\circ] \rightarrow S[\circ\circ/D]\ D[]$
$(\overset{s}{x}/T)\ T \rightarrow \overset{s}{x}$	$S[\circ\circ] \rightarrow S[\circ\circ/T]\ T[]$
$A\ (\overset{s}{x}\backslash A) \rightarrow \overset{s}{x}$	$S[\circ\circ] \rightarrow A[]\ S[\circ\circ\backslash A]$
$B\ (\overset{s}{x}\backslash B) \rightarrow \overset{s}{x}$	$S[\circ\circ] \rightarrow B[]\ S[\circ\circ\backslash B]$
$D\ (\overset{s}{x}\backslash D) \rightarrow \overset{s}{x}$	$S[\circ\circ] \rightarrow D[]\ S[\circ\circ\backslash D]$
$T\ (\overset{s}{x}\backslash T) \rightarrow \overset{s}{x}$	$S[\circ\circ] \rightarrow T[]\ S[\circ\circ\backslash T]$

3.2 lil $\subseteq$ hl

This proof is derived from the standard technique used in the simulation of pushdown automata by context-free grammars. In that proof nonterminals are triples $\langle p, q, l\rangle$ where p is the state of the machine when the stack symbol l is placed on top of the pushdown and q is the state of the machine at the point that l is eventually removed. This nonterminal derives the terminal string w if w is the string read by the machine between these two points in the computation. In our proof nonterminals of the lig take the place of states and we have nonterminals in the hg that are triples of the form $\langle A, B, l\rangle$. Figure 4 shows a derivation in which $B[]$ is the distinguished descendent of $A[l]$ and the stack symbol l has been popped. The terminal yield between these two points can be expressed as a pair $w_1 \uparrow w_2$ and by wrapping this around the yield of $B[]$ (the string u), the string derived from $A[l]$ is obtained (the string $w_1 u w_2$).

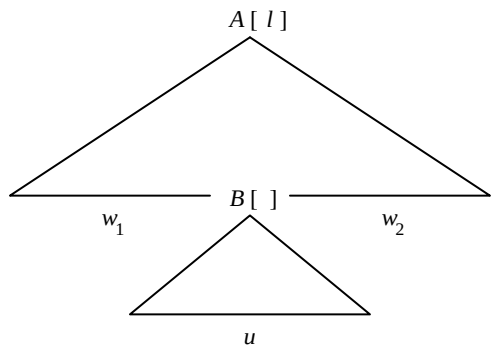

Figure 4: l is popped from stack

Let $G = (V_N, V_T, V_I, S, P)$ be a lig where, without loss of generality, we assume that productions in P have the following forms, where $A, A_1, \ldots, A_n \in V_N$, $l \in V_I$ and $w \in V_T^\epsilon$.

- $A[\circ\circ] \rightarrow A_1[] \ldots A_{i-1}[]A_i[\circ\circ\, l]A_{i+1}[] \ldots A_n[]$.
- $A[\circ\circ\, l] \rightarrow A_1[] \ldots A_{i-1}[]A_i[\circ\circ]A_{i+1}[] \ldots A_n[]$.
- $A[] \rightarrow w$.

We construct a hg, $G' = (V'_N, V_T, S, P')$, where

$$V'_N = V_N \cup \{\, (A_1, A_2, \eta) \mid A_1, A_2 \in V_N \text{ and } \eta \in V_I \cup \{\, \epsilon \,\} \,\}$$

and P' is as follows.

- If $A[\circ\circ] \rightarrow A_1[] \ldots A_{i-1}[]A_i[\circ\circ\, l]A_{i+1}[] \ldots A_n[] \in P$ then for all $B \in V_N$ let

$$(A, B, \epsilon) \rightarrow C_{i,n}(A_1, \ldots, A_{i-1}, (A_i, B, l), A_{i+1}, \ldots, A_n) \in P'$$

- If $A[\circ\circ\, l] \rightarrow A_1[] \ldots A_{i-1}[]A_i[\circ\circ]A_{i+1}[] \ldots A_n[] \in P$ then for all $B \in V_N$ let

$$(A, B, l) \rightarrow C_{i,n}(A_1, \ldots, A_{i-1}, (A_i, B, \epsilon), A_{i+1}, \ldots, A_n) \in P'$$

- If $A[] \rightarrow w \in P$ then let $A \rightarrow C_{1,1}(\epsilon \uparrow w) \in P'$

- For every $A, B, C \in V_N$, and $\eta \in V_I \cup \{\epsilon\}$ let

$$(A, B, \eta) \rightarrow W((A, C, \epsilon), (C, B, \eta)) \in P'$$

$$(A, A, \epsilon) \rightarrow C_{1,1}(\epsilon_{\uparrow}\epsilon) \in P'$$

$$A \rightarrow W((A, B, \epsilon), B) \in P'$$

To show that $L(G) = L(G')$ we prove Lemma 3.4.

Lemma 3.4 Both of the following two statements hold.

Part 1 For all $A, B \in V_N$, $\eta \in V_I \cup \{\epsilon\}$ and $w_1, w_2 \in V_T^*$

$$A[\eta] \underset{G}{\overset{*}{\Longrightarrow}} w_1 B[] w_2 \text{ where } B[] \text{ is a distinguished descendent of } A[\eta]$$

if and only if

$$(A, B, \eta) \underset{G'}{\Longrightarrow} w_{1\uparrow} w_2$$

Part 2 For all $A \in V_N$ and $w \in V_T^*$ $A[] \underset{G}{\overset{*}{\Longrightarrow}} w$ if and only if $w = w_1 w_2$ and $A \underset{G'}{\Longrightarrow} w_{1\uparrow} w_2$ for some w_1 and w_2.

Proof We prove this by simultaneously inducting on the length of derivations in both parts of the lemma. There are four implications to consider.

Basis:

Part 1 *only if* Consider the zero-step derivation $A[\eta] \underset{G}{\Longrightarrow} w_1 B[] w_2$ where $A = B$, $\eta = \epsilon$, and $w_1 = w_2 = \epsilon$. Using the production $(A, A, \epsilon) \rightarrow C_{1,1}(\epsilon_{\uparrow}\epsilon)$ of P' we have the desired derivation in G'.

Part 1 *if* The only one-step derivation of the appropriate form is $(A, A, \epsilon) \underset{G'}{\Longrightarrow} \epsilon_{\uparrow}\epsilon$ which involves use of the production $(A, A, \epsilon) \rightarrow C_{1,1}(\epsilon_{\uparrow}\epsilon) \in P'$. Since $\underset{G}{\overset{*}{\Longrightarrow}}$ is reflexive we have $A[] \underset{G}{\overset{*}{\Longrightarrow}} A[]$.

Part 2 $A[] \underset{G}{\Longrightarrow} w$ if and only if $w \in V_T^{\epsilon}$ and $A[] \rightarrow w \in P$ if and only if $A \rightarrow C_{1,1}(\epsilon_{\uparrow} w) \in P'$ if and only if $A \underset{G'}{\Longrightarrow} \epsilon_{\uparrow} w$.

Inductive step:

Part 1 *only if* Suppose that $A[\eta] \underset{G}{\overset{k}{\Longrightarrow}} w_1 B[] w_2$ where $B[]$ is a distinguished descendent of $A[\eta]$.

- If the derivation begins with the production

$$A[\circ\circ] \rightarrow A_1[] \ldots A_{i-1}[] A_i[\circ\circ\, l] A_{i+1}[] \ldots A_n[]$$

then it can be decomposed as follows.

$$\begin{aligned} A[\eta] &\underset{G}{\Longrightarrow} A_1[]\ldots A_{i-1}[]A_i[\eta l]A_{i+1}[]\ldots A_n[] \\ &\overset{k_1}{\underset{G}{\Longrightarrow}} u_A A_i[\eta l]v_A \\ &\overset{k_2}{\underset{G}{\Longrightarrow}} u_A u_i B'[\eta]v_i v_A \\ &\overset{k_3}{\underset{G}{\Longrightarrow}} u_A u_i u_{B'} B[]v_{B'} v_i v_A \\ &= w_1 B[]w_2 \end{aligned}$$

where k_1, k_2, and k_3 are less than k and for each $1 \leq j \leq n$ such that $j \neq i$ we have $A_j[] \overset{k'_j}{\underset{G}{\Longrightarrow}} w_{A_j}$ where $k_j < k$ and $u_A = w_{A_1}\ldots w_{A_{i-1}}$ and $v_A = w_{A_{i+1}}\ldots w_{A_n}$, and we have the following derivations

$$A_i[l] \overset{k_1+k_2}{\underset{G}{\Longrightarrow}} u_i B'[]v_i \qquad \text{and} \qquad B'[\eta] \overset{k_3}{\underset{G}{\Longrightarrow}} u_{B'} B[]v_{B'}$$

By the inductive hypothesis we have

$$(A_i, B', l) \underset{G'}{\Longrightarrow} u_{i\uparrow} v_i, \qquad (B', B, \eta) \underset{G'}{\Longrightarrow} u_{B'\uparrow} v_{B'} \qquad \text{and} \qquad A_j \underset{G'}{\Longrightarrow} u_{j\uparrow} v_j$$

for each $1 \leq j \leq n$ such that $j \neq i$ and $w_{A_j} = u_j v_j$.
By the construction we know that the following productions are in P'.

$$(A, B', \epsilon) \rightarrow C_{i,n}(A_1, \ldots, A_{i-1}, (A_i, B', l), A_{i+1}, \ldots, A_n)$$

$$(A, B, \eta) \rightarrow W((A, B', \epsilon), (B', B, \eta))$$

Then we have

$$\begin{aligned} (A, B', \epsilon) &\underset{G'}{\Longrightarrow} C_{i,n}(u_{1\uparrow}v_1, \ldots, u_{i-1\uparrow}v_{i-1}, u_{i\uparrow}v_i, u_{i+1\uparrow}v_{i+1}, \ldots, u_{n\uparrow}v_n) \\ &= u_1 v_1 \ldots u_{i-1} v_{i-1} u_{i\uparrow} v_i u_{i+1} v_{i+1} \ldots u_n v_n \\ \text{thus,}\quad (A, B, \eta) &\underset{G'}{\Longrightarrow} W(u_1 v_1 \ldots u_{i-1} v_{i-1} u_{i\uparrow} v_i u_{i+1} v_{i+1} \ldots u_n v_n, u_{B'\uparrow} v_{B'}) \\ &= u_1 v_1 \ldots u_{i-1} v_{i-1} u_i u_{B'\uparrow} v_{B'} v_i u_{i+1} v_{i+1} \ldots u_n v_n \\ &= w_{1\uparrow} w_2 \end{aligned}$$

- Alternatively, suppose that the derivation begins with the production

$$A[\circ\circ\, l] \rightarrow A_1[]\ldots A_{i-1}[]A_i[\circ\circ]A_{i+1}[]\ldots A_n[]$$

In this case we know that $\eta = l$ and the derivation can be decomposed as follows.

$$\begin{aligned} A[l] &\underset{G}{\Longrightarrow} A_1[]\ldots A_{i-1}[]A_i[]A_{i+1}[]\ldots A_n[] \\ &\overset{*}{\underset{G}{\Longrightarrow}} u_A A_i[]v_A \\ &\overset{*}{\underset{G}{\Longrightarrow}} u_A u_i B[]v_i v_A \\ &= w_1 B[]w_2 \end{aligned}$$

where for each $1 \leq j \leq n$ such that $j \neq i$ we have $A_j[] \overset{*}{\underset{G}{\Longrightarrow}} w_{A_j}$ where $u_A = w_{A_1}\ldots w_{A_{i-1}}$ and $v_A = w_{A_{i+1}}\ldots w_{A_n}$, and we have $A_i[] \overset{*}{\underset{G}{\Longrightarrow}} u_i B[]v_i$. By the inductive hypothesis we have

$$(A_i, B, \epsilon) \underset{G'}{\Longrightarrow} u_{i\uparrow} v_i \qquad \text{and} \qquad A_j \underset{G'}{\Longrightarrow} u_{j\uparrow} v_j$$

for each $1 \leq j \leq n$ such that $j \neq i$ and $w_{A_j} = u_j v_j$.
By the construction we know that the following production is in P'.

$$(A, B, l) \rightarrow C_{i,n}(A_1, \ldots, A_{i-1}, (A_i, B, \epsilon), A_{i+1}, \ldots, A_n)$$

Hence, we have the following derivation.

$$(A, B, l) \underset{G'}{\Longrightarrow} C_{i,n}(u_{1\uparrow} v_1, \ldots, u_{n\uparrow} v_n) = w_{1\uparrow} w_2$$

Part 1 *if* Suppose that $(A, B, \eta) \overset{k}{\underset{G'}{\Longrightarrow}} w_{1\uparrow} w_2$.

- Suppose that $\eta = \epsilon$ and this derivation begins with use of the production

$$(A, B, \epsilon) \rightarrow C_{i,n}(A_1, \ldots, A_{i-1}, (A_i, B, l), A_{i+1}, \ldots, A_n)$$

and that $w_{1\uparrow} w_2 = u_1 v_1 \ldots u_{i\uparrow} v_i \ldots u_n v_n$ where $(A_i, B, l) \overset{k_i}{\underset{G'}{\Longrightarrow}} u_{i\uparrow} v_i$ and for each $1 \leq j \leq n$ such that $j \neq i$ we have $A_j \overset{k_j}{\underset{G'}{\Longrightarrow}} u_{j\uparrow} v_j$ where $k_1, \ldots, k_n$ are all less than k.
Thus, by induction we know that for each $1 \leq j \leq n$ such that $j \neq i$ we have the derivation $A_j[] \overset{*}{\underset{G}{\Longrightarrow}} u_j v_j$ and $A_i[l] \overset{*}{\underset{G}{\Longrightarrow}} u_i B[] v_i$. By the construction we know that P contains the following production.

$$A[\circ\circ] \rightarrow A_1[] \ldots A_{i-1}[] A_i[\circ\circ\, l] A_{i+1}[] \ldots A_n[]$$

Thus, we have the following derivation in G.

$$\begin{aligned} A[] &\underset{G}{\Longrightarrow} A_1[] \ldots A_{i-1}[] A_i[l] A_{i+1}[] \ldots A_n[] \\ &\overset{*}{\underset{G}{\Longrightarrow}} u_1 v_1 \ldots u_{i-1} v_{i-1} u_i B[] v_i u_{i+1} v_{i+1} \ldots u_n v_n \\ &= w_1 B[] w_2 \end{aligned}$$

- Alternatively, suppose that $\eta \in V_I$ and this derivation begins with use of the production $(A, B, \eta) \rightarrow C_{i,n}(A_1, \ldots, A_{i-1}, (A_i, B, \epsilon), A_{i+1}, \ldots, A_n)$ and that

$$w_{1\uparrow} w_2 = u_1 v_1 \ldots u_{i\uparrow} v_i \ldots u_n v_n$$

where for each $1 \leq j \leq n$ such that $j \neq i$ we have $A_j \overset{k_j}{\underset{G'}{\Longrightarrow}} u_{j\uparrow} v_j$ and $(A_i, B, \epsilon) \overset{k_i}{\underset{G'}{\Longrightarrow}} u_{i\uparrow} v_i$ where each k_j is less than k $(1 \leq j \leq n)$.
Thus, by induction we know that for each $1 \leq j \leq n$ such that $j \neq i$ we have the derivation $A_j[] \overset{*}{\underset{G}{\Longrightarrow}} u_j v_j$ as well as the derivation $A_i[] \overset{*}{\underset{G}{\Longrightarrow}} u_i B[] v_i$. By the construction we know that P contains the following production.

$$A[\circ\circ\, \eta] \rightarrow A_1[] \ldots A_{i-1}[] A_i[\circ\circ] A_{i+1}[] \ldots A_n[]$$

Thus, we have the following derivation in G.

$$\begin{aligned} A[\eta] &\underset{G}{\Longrightarrow} A_1[] \ldots A_{i-1}[] A_i[] A_{i+1}[] \ldots A_n[] \\ &\overset{*}{\underset{G}{\Longrightarrow}} u_1 v_1 \ldots u_{i-1} v_{i-1} A_i[] u_{i+1} v_{i+1} \ldots u_n v_n \\ &\overset{*}{\underset{G}{\Longrightarrow}} u_1 v_1 \ldots u_{i-1} v_{i-1} u_i B[] v_i u_{i+1} v_{i+1} \ldots u_n v_n \\ &= w_1 B[] w_2 \end{aligned}$$

- Alternatively, suppose that the derivation begins with use of the production

$$(A, B, \eta) \rightarrow W((A, C, \epsilon), (C, B, \eta))$$

where $\eta \in V_I \cup \{\epsilon\}$, $w_{1\uparrow}w_2 = u_1u_{2\uparrow}v_2v_1$ such that $(A, C, \epsilon) \overset{k_1}{\underset{G'}{\Longrightarrow}} u_{1\uparrow}v_1$ and $(C, B, \eta) \overset{k_2}{\underset{G'}{\Longrightarrow}} u_{2\uparrow}v_2$ where k_1 and k_2 are less than k.

Thus, by induction we know that $A[] \overset{*}{\underset{G}{\Longrightarrow}} u_1C[]v_1$ and $C[\eta] \overset{*}{\underset{G}{\Longrightarrow}} u_2B[]v_2$. Thus, we have the following derivation in G.

$$\begin{aligned} A[\eta] &\underset{G}{\Longrightarrow} u_1C[\eta]v_1 \\ &\overset{*}{\underset{G}{\Longrightarrow}} u_1u_2B[]v_2v_1 \\ &= w_1B[]w_2 \end{aligned}$$

Part 2 *only if* Suppose that $A[] \overset{k}{\underset{G}{\Longrightarrow}} w$. In this case we have the following derivation.

$$\begin{aligned} A[] &\underset{G}{\Longrightarrow} A_1[]\ldots A_{i-1}[]A_i[l]A_{i+1}[]\ldots A_n[] \\ &\overset{k_1}{\underset{G}{\Longrightarrow}} u_AA_i[l]v_A \\ &\overset{k_2}{\underset{G}{\Longrightarrow}} u_Au_iB[]v_iv_A \\ &\overset{k_3}{\underset{G}{\Longrightarrow}} u_Au_iw_Bv_iv_A \\ &= w \end{aligned}$$

where k_1, k_2, and k_3 are less than k and in the subderivation $A_i[l] \overset{k_2}{\underset{G}{\Longrightarrow}} u_iB[]v_i$ $B[]$ is a distinguished descendent of $A_i[l]$; furthermore $B[] \overset{k_3}{\underset{G}{\Longrightarrow}} w_B$. Thus, by induction we know that $(A_i, B, l) \underset{G'}{\Longrightarrow} u_{i\uparrow}v_i$, $B \underset{G'}{\Longrightarrow} u_{B\uparrow}v_B$ where $u_Bv_B = w_B$, and that for each $1 \leq j \leq n$ such that $j \neq i$ we have the derivation $A_j \underset{G'}{\Longrightarrow} u_{j\uparrow}v_j$ such that $u_1v_1\ldots u_{i-1}v_{i-1} = u_A$ and $u_{i+1}v_{i+1}\ldots u_nv_n = v_A$. From the construction we have the following productions in P'.

$$(A, B, \epsilon) \rightarrow C_{i,n}(A_1, \ldots, A_{i-1}, (A_i, B, l), A_{i+1}, \ldots, A_n)$$
$$A \rightarrow W((A, B, \epsilon), B)$$

Hence, we have the following.

$$\begin{aligned} (A, B, \epsilon) &\underset{G'}{\Longrightarrow} C_{i,n}(u_{1\uparrow}v_1, \ldots, u_{i-1\uparrow}v_{i-1}, u_{i\uparrow}v_i, u_{i+1\uparrow}v_{i+1}\ldots, u_{n\uparrow}v_n) \\ &= u_1v_1\ldots u_{i-1}v_{i-1}u_{i\uparrow}v_iu_{i+1}v_{i+1}\ldots u_nv_n \\ \text{thus,} \qquad A &\underset{G'}{\Longrightarrow} W(u_1v_1\ldots u_{i-1}v_{i-1}u_{i\uparrow}v_iu_{i+1}v_{i+1}\ldots u_nv_n, u_{B\uparrow}v_B) \\ &= u_1v_1\ldots u_{i-1}v_{i-1}u_iu_{B\uparrow}v_Bv_iu_{i+1}v_{i+1}\ldots u_nv_n \\ &= u_Au_iu_{B\uparrow}v_Bv_iv_A \end{aligned}$$

where $w = u_Au_iu_Bv_Bv_iv_A$.

Part 2 *if* Suppose that $A \overset{k}{\underset{G'}{\Longrightarrow}} w_{1\uparrow}w_2$. This derivation begins with the production $A \rightarrow W((A, B, \epsilon), B)$ such that $w_{1\uparrow}w_2 = u_1u_{2\uparrow}v_2v_1$, $(A, B, \epsilon) \overset{k_1}{\underset{G'}{\Longrightarrow}} u_{1\uparrow}v_1$ and $B \overset{k_2}{\underset{G'}{\Longrightarrow}} u_{2\uparrow}v_2$ where k_1 and k_2 are less than k.

Thus, by induction we know that

$$A[] \overset{*}{\underset{G}{\Longrightarrow}} u_1B[]v_1 \overset{*}{\underset{G}{\Longrightarrow}} u_1u_2v_2v_1$$

∎

3.3 hl$\subseteq$ tal

The wrapping operation of hg can be simulated by the adjunction operation of tag. Suppose we have $B \underset{G}{\Longrightarrow} u_{1\uparrow}v_1$, $C \underset{G}{\Longrightarrow} u_{2\uparrow}v_2$ and we have the production $A \rightarrow W(B,C)$. From this we have $A \underset{G}{\Longrightarrow} u_1u_{2\uparrow}u_2v_1$. This can be simulated by a tag as follows. Assume that corresponding to the derivations $B \underset{G}{\Longrightarrow} u_{1\uparrow}v_1$ and $C \underset{G}{\Longrightarrow} u_{2\uparrow}v_2$ we have trees β_1 and β_2 such that $yield\,(\beta_1) = u_1Bv_1$ and $yield\,(\beta_2) = u_2Cv_2$. Corresponding to the production $A \rightarrow W(B,C)$ include a tree in the grammar in which β_1 can be adjoined at the parent of a node at which β_2 can be adjoined; then we can derive a tree γ such that $yield\,(\gamma) = u_1u_2Av_2v_1$.

From a hg, $G = (V_N, V_T, S, P)$, we construct a tag, $G' = (V_N, V_T, S, V_L, \mathcal{I}, \mathcal{A}, _)$, such that $L(G) = L(G')$. Without loss of generality, we assume that the productions in P are of the following form where $A, B, C \in V_N$, $w_1, w_2 \in V_T^{\epsilon}$ and $f \in \{\, C_{1,2}, C_{2,2}, W \,\}$.

$$A \rightarrow f(B,C) \qquad A \rightarrow C_{1,1}(w_{1\uparrow}w_2)$$

The elementary trees of G' are given as follows.

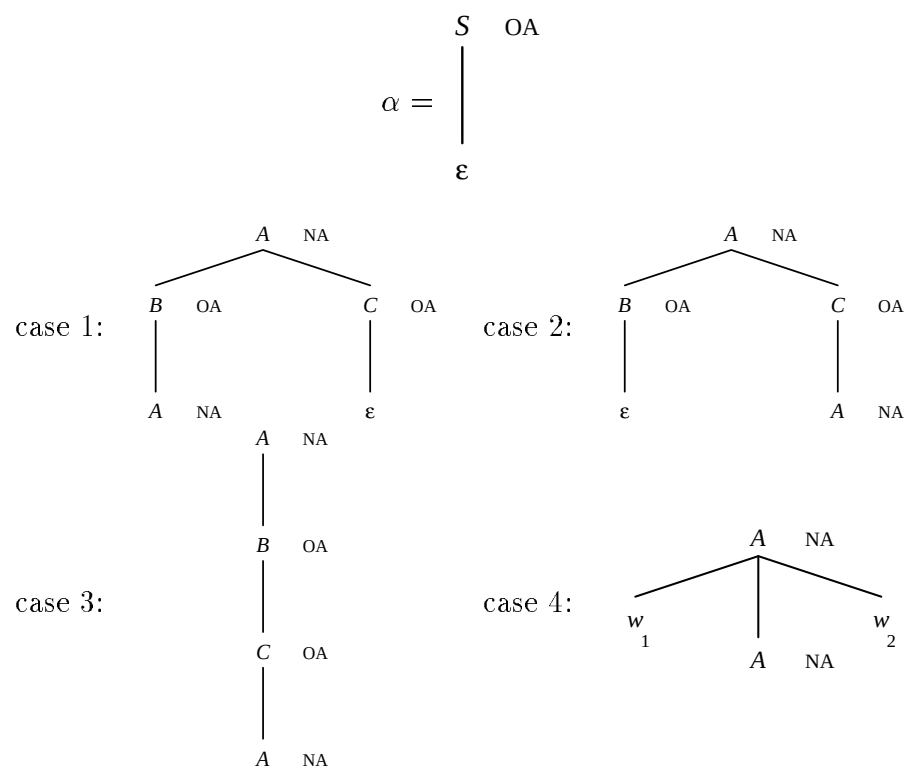

Figure 5: Trees in G'

$\mathcal{I}$ contains the single initial tree α shown in Figure 5. The set, $\mathcal{A}$, of auxiliary trees is as follows.

Case 1: If $A \rightarrow C_{1,2}(B,C) \in P$ then include the tree shown in Figure 5 for case 1 in $\mathcal{A}$

Case 2: If $A \rightarrow C_{2,2}(B,C) \in P$ then include the tree shown in Figure 5 for case 2 in $\mathcal{A}$

Case 3: If $A \rightarrow W(B,C) \in P$ then include the tree shown in Figure 5 for case 3 in $\mathcal{A}$

Case 4: If $A \rightarrow C_{1,1}(w_{1\uparrow}w_2) \in P$ then include the tree shown in Figure 5 for case 4 in $\mathcal{A}$

Let V_L and $_$ be such that V_L is the set of tree labels $\left\{ \overline{\beta_1}, \ldots, \overline{\beta_n} \right\}$ where $\mathcal{A} = \{ \beta_1, \ldots, \beta_n \}$.

G' simulates derivations of G such that whenever a production is used in G the corresponding tree is used in G'. To show that $L(G) = L(G')$ we prove the following lemma by induction.

Lemma 3.5 For all $A \in V_N$, $w_1, w_2 \in V_T^*$ $A \underset{G}{\Longrightarrow} w_{1\uparrow} w_2$ if and only if there exist β, β' such that $\beta \overset{*}{\underset{G'}{\Longrightarrow}} \beta'$, $\beta \in \mathcal{A} \cap aux\,(V_N, V_T, V_L, A)$, β has no OA nodes and $yield\,(\beta') = w_1 A w_2$.

Proof The basis of the induction involves proving the above lemma for derivations in G involving the use of one production and derivations in G' involving the adjunction of one tree. Thus, the only relevant part of the construction is case 4 and the desired equivalence clearly holds.

For the inductive step proving each direction of the equivalence involves consideration of cases 1–3 in the above construction. The details are straightforward, so rather than enumerating all three cases for each direction we show one case for each direction.

Suppose that $A \overset{k}{\underset{G}{\Longrightarrow}} w_{1\uparrow} w_2$ for some $A \in V_N$ involving use of the production $A \rightarrow C_{1,2}(B, C)$ and the subderivations $B \overset{k_1}{\underset{G}{\Longrightarrow}} u_{1\uparrow} u_2$ and $C \overset{k_2}{\underset{G}{\Longrightarrow}} v_{1\uparrow} v_2$ where k_1 and k_2 are less than k and $C_{1,2}(u_{1\uparrow} u_2, v_{1\uparrow} v_2) = w_{1\uparrow} w_2$. Thus, by the induction hypothesis we know that $\beta_1 \overset{*}{\underset{G'}{\Longrightarrow}} \beta_1'$ for some $\beta_1 \in \mathcal{A} \cap aux\,(V_N, V_T, V_L, B)$ such that $yield\,(\beta_1') = u_1 B u_2$ and $\beta_2 \overset{*}{\underset{G'}{\Longrightarrow}} \beta_2'$ for some $\beta_2 \in \mathcal{A} \cap aux\,(V_N, V_T, V_L, C)$ such that $yield\,(\beta_2') = v_1 C v_2$. Let β be the tree introduced in case 1 of the construction due to the presence of $A \rightarrow C_{1,2}(B, C)$ in P. We have $\beta \overset{*}{\underset{G'}{\Longrightarrow}} \nabla\,(\nabla\,(\beta, \beta_1', 1)\,, \beta_2', 2)$ where $yield\,(\nabla\,(\nabla\,(\beta, \beta_1', 1)\,, \beta_2', 2)) = u_1 A u_2 v_1 v_2$.

Suppose that $\beta \overset{k}{\underset{G'}{\Longrightarrow}} \beta'$ for some $\beta \in \mathcal{A} \cap aux\,(V_N, V_T, A)$. Suppose that β was introduced in case 3 of the construction and that $\beta' = \nabla\,(\nabla\,(\beta, \beta_2, 11)\,, \beta_1, 1)$ where $\beta_1' \overset{k_1}{\underset{G'}{\Longrightarrow}} \beta_1$, for some $\beta_1' \in \mathcal{A} \cap aux\,(V_N, V_T, B)$, and $\beta_2' \overset{k_2}{\underset{G'}{\Longrightarrow}} \beta_2$ for some $\beta_2' \in \mathcal{A} \cap aux\,(V_N, V_T, C)$ where k_1 and k_2 are less than k. Let $yield\,(\beta_1) = u_1 B u_2$ and $yield\,(\beta_2) = v_1 C v_2$. Thus $yield\,(\beta') = u_1 v_1 A v_2 u_2$. Thus, by induction we know that $B \underset{G}{\Longrightarrow} u_{1\uparrow} u_2$ and $C \underset{G}{\Longrightarrow} v_{1\uparrow} v_2$. Since P contains the production $A \rightarrow W(B, C)$ we have $A \underset{G}{\Longrightarrow} W(u_{1\uparrow} u_2, v_{1\uparrow} v_2) = u_1 v_{1\uparrow} v_2 u_2$. ∎

Example 3.2 Consider the hg, $G = (\{ S, T_a, T_b, A, B \}\,, \{ a, b \}\,, S, P)$, that generates the language $\{ ww \mid w \in \{ a, b \}^* \}$ where P is as follows.

$$P = \left\{ \begin{array}{c} S \rightarrow C_{1,1}(\epsilon_\uparrow \epsilon), \quad S \rightarrow C_{2,2}(A, T_a), \quad S \rightarrow C_{2,2}(B, T_b), \\ T_a \rightarrow W(S, A), \quad T_b \rightarrow W(S, B), \quad A \rightarrow C_{1,1}(\epsilon_\uparrow a), \quad B \rightarrow C_{1,1}(\epsilon_\uparrow b) \end{array} \right\}$$

The following tag, G would be produced by the above construction

$$G' = (\{ S, T_a, T_b, A, B \}\,, \left\{ \overline{\beta_1}, \ldots, \overline{\beta_7} \right\}, \{ a, b \}\,, S, \{ \alpha \}\,, \mathcal{A}, _)$$

where α is as given in the construction and the trees in $\mathcal{A}$ are shown in Figure 6.

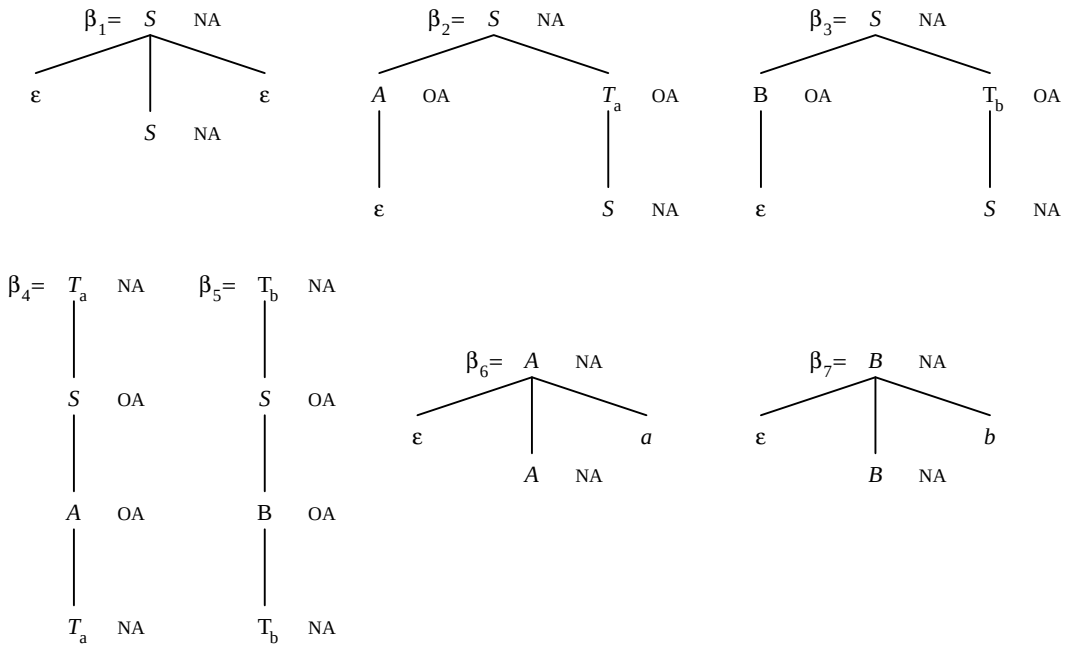

Figure 6: Auxiliary trees of G'

3.4 tal $\subseteq$ ccl

Consider a tag derivation in which trees are adjoined at the deepest adjunction points first. Figure 7 illustrates an adjunction at a node labelled B. Adjunctions have been completed on the path marked with the solid line whereas adjunction at nodes on the path marked with the broken line have yet to be made. The broken line can be viewed as a stack of nodes and the effect of adjoining a tree can be viewed as "pushing" a new path (the spine of the adjoined tree) onto the top of the stack. The stacking of paths seen in adjunction can be simulated by ccg

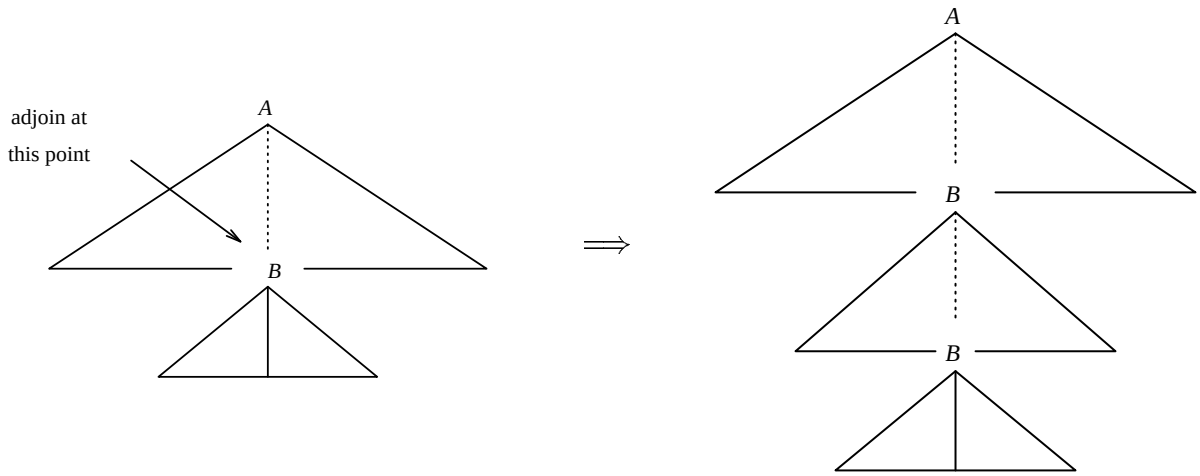

Figure 7: Stacking of paths

composition rules. To do this we encode auxiliary trees with ccg categories. This is achieved by arranging that the auxiliary trees are pruned. Informally, an auxiliary tree is pruned if it is at most binary branching and siblings of nodes on the spine have OA constraints and have a single child labelled by ϵ. An example of such a tree is shown in Figure 8. Trees that are pruned can be encoded by ccg categories, adjunction at the spine simulated by composition

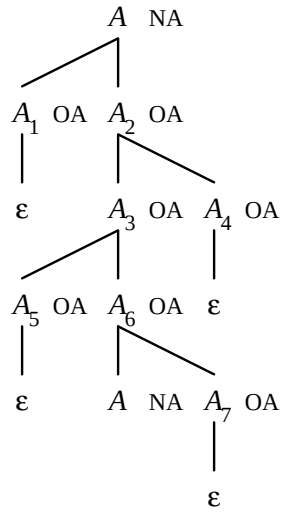

Figure 8: Example of a pruned tree

rules (instances of the schemata F_n and B_n where $n \geq 1$) and adjunction at siblings of the spine simulated by application rules (instances of the schemata F_0 and B_0).

In the proof we use a result given in Appendix A showing that an arbitrary tag can be converted into an equivalent tag, $G = (V_N, V_T, V_L, S, \{\alpha\}, \mathcal{A}_1 \cup \mathcal{A}_2, \text{-})$, such that α is shown in Figure 5 and the two sets that make up the auxiliary trees are as follows. $\mathcal{A}_1$ contains the trees that introduce terminal symbols. Each auxiliary tree $\beta \in \mathcal{A}_1$ has the form of the tree shown in Figure 9 where $A \in V_N$ and $w \in V_T^{\epsilon}$.

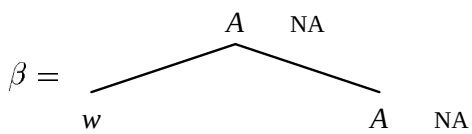

Figure 9: Form of trees in $\mathcal{A}_1$

Each tree $\beta \in \mathcal{A}_2$ has the property that $\langle \beta, ft(\beta) \rangle$ is *pruned* where $\langle \beta, d \rangle$ is *pruned* if and only if the following conditions hold.

- $\beta \in aux(V_N, V_T, V_L)$ and $d \in spine(\beta)$.
- $d'3 \notin dom(\beta)$ for each d' that is a proper prefix of d, i.e., there is at most binary branching.
- For each d' that is a prefix of d either $\beta(d') = \langle A, \emptyset, \mathsf{false} \rangle$ for some $A \in V_N$ or $\beta(d') = \langle A, V_L, \mathsf{true} \rangle$, i.e., either adjunction is obligatory with any tree with root labelled by the nonterminal A, or adjunction is not allowed.
- For each d' that is a proper prefix of d let $i = 1$ if $d'1$ is not a prefix of d, otherwise $i = 2$. $\beta(d'i) = \langle A, V_L, \mathsf{true} \rangle$, $\beta(d'i1) = \epsilon$ and $\beta(d'i2)$ is undefined for some $A \in V_N$, i.e., each sibling of a node on the path from the root to address d of β is labelled by a member of V_N, has an OA constraint with no restriction on which trees can be adjoined, and has a single child labelled with ϵ.

We make the further assumption, satisfied by the construction in Appendix A, that none of the trees in $\mathcal{A}_1$ can be adjoined at nodes on the spine of trees in $\mathcal{A}_2$.

From this tag, G, we construct an equivalent ccg, $G' = (V'_N, V_T, S, f, R)$, where f and R are defined below and $V'_N = V_N \cup \left\{ \hat{A} \mid A \in V_N \right\}$. First we define an encoding function $enc : aux\,(V_N, V_T, V_L) \times \mathcal{N}^*_+ \rightarrow cat\,(V'_N)$ with which each $\langle \beta, d \rangle$ that is pruned is encoded by a member of $cat\,(V'_N)$. Let $\beta \in aux\,(V_N, V_T, V_L)$.

- If $\beta(\epsilon) = \langle A, \emptyset, \mathsf{false} \rangle$ then $enc\,(\beta, \epsilon) = A$. If $\beta(\epsilon) = \langle A, V_L, \mathsf{true} \rangle$ then $enc\,(\beta, \epsilon) = (A/\hat{A})$.
- Let d be a proper prefix of $ft\,(\beta)$ and $enc\,(\beta, d) = c$
  - if $d1 \in spine\,(\beta)$ and $\beta(d2) = \langle C, V_L, \mathsf{true} \rangle$ then if $\beta(d1) = \langle B, V_L, \mathsf{true} \rangle$ then $enc\,(\beta, d1) = ((c/C)/\hat{B})$ and if $\beta(d1) = \langle B, \emptyset, \mathsf{false} \rangle$ then $enc\,(\beta, d1) = (c/C)$.
  - if $d2 \in spine\,(\beta)$ and $\beta(d1) = \langle B, V_L, \mathsf{true} \rangle$ then if $\beta(d2) = \langle C, V_L, \mathsf{true} \rangle$ then $enc\,(\beta, d2) = ((c\backslash B)/\hat{C})$ and if $\beta(d2) = \langle C, \emptyset, \mathsf{false} \rangle$ then $enc\,(\beta, d2) = (c\backslash B)$.
  - If $d2 \notin dom\,(\beta)$ and $d1 \in dom\,(\beta)$ then if $\beta(d1) = \langle B, V_L, \mathsf{true} \rangle$ then $enc\,(\beta, d1) = (c/\hat{B})$ and if $\beta(d1) = \langle B, \emptyset, \mathsf{false} \rangle$ then $enc\,(\beta, d1) = c$.

The function f of G' is defined as follows.

- For each $\beta \in \mathcal{A}_1$ such that $\beta(\epsilon) = \langle A, \emptyset, \mathsf{false} \rangle$ and $\beta(1) = w$ where $A \in V_N$ and $w \in V_T^\epsilon$ include A in $f(w)$.
- For each $\beta \in \mathcal{A}_2$ include c and $\hat{c}$ in $f(\epsilon)$ where $enc\,(\beta, ft\,(\beta)) = c$ and $\hat{c}$ is identical to c with the exception of the target category[3].

The tree in Figure 8 would lead to the following.

$$((((((((A\backslash A_1)/\hat{A}_2)/A_4)/\hat{A}_3)\backslash A_5)/\hat{A}_6)/A_7) \in f(\epsilon)$$

$$((((((((\hat{A}\backslash A_1)/\hat{A}_2)/A_4)/\hat{A}_3)\backslash A_5)/\hat{A}_6)/A_7) \in f(\epsilon)$$

We complete the construction of G' by defining the set of combinatory rules in R. Let $k = max\,(\{\, arity\,(c) \,|\, c \in f(w) \text{ for some } w \in V_T^\epsilon \,\})$. For each $A \in V_N$, each $0 \leq i \leq k$, and $|_1, \ldots, |_i \in \{\, /, \backslash \,\}$ include the following rules in R.

- $(x/A)\ A \rightarrow x$
- $A\ (x\backslash A) \rightarrow x$
- $(x/\hat{A})\ (\ldots(\hat{A}|_1 z_1)|_2 \ldots |_i z_i) \rightarrow (\ldots(x|_1 z_1)|_2 \ldots |_i z_i)$

The combinatory rules in R permit composition (when $i > 0$) only when the target category of the secondary component is of the form $\hat{A}$. This corresponds to adjunction with a tree in $\mathcal{A}_2$ at a node on the spine. In the above encoding of trees by categories every symbol of the form $\hat{A}$ is preceeded by a forward slash. Therefore, we only need forward composition rules.

Before proving the equivalence of G and G' we define a new derivation order for the above tag, G. For each tree $\beta \in aux\,(V_N, V_T, V_L)$ we define $next\,(\beta)$ which gives the address at which the next adjunction should occur. If the deepest OA node in β is not on the spine then this is the next adjunction point, otherwise it is the deepest OA node on the spine. Note that $next\,(\beta)$ is not defined when there are several deepest OA nodes not on the spine. More formally, $next\,(\beta) = d$ if $d \in dom\,(\beta)$ and one or other of the following holds.

[3]Formally, $(c_1\hat{/}c_2) = (\hat{c_1}/c_2)$ and $(c_1\hat{\backslash}c_2) = (\hat{c_1}\backslash c_2)$.

- $d \notin spine\,(\beta)$, $\beta(d) = \langle A, V_L, \mathsf{true}\rangle$ for some $A \in V_N$ and there is no $d' \in dom\,(\beta)$ such that $\beta(d') = \langle B, V_L, \mathsf{true}\rangle$ for some $B \in V_N$ and $|d'| \geq |d|$. We use $|d|$ to denote the length of the address $d \in \mathcal{N}_+^*$. This gives the depth of a node with that address.

- $d \in spine\,(\beta)$, $\beta(d) = \langle A, V_L, \mathsf{true}\rangle$ for some $A \in V_N$ and there is no $d' \in dom\,(\beta)$ such that $\beta(d') = \langle B, V_L, \mathsf{true}\rangle$ for some $B \in V_N$ and $|d'| > |d|$

Let the set $T_k(G)$ include trees derived in k or fewer steps, for $k \geq 0$, as follows.

- $T_0(G) = \mathcal{A}_1 \cup \mathcal{A}_2$

- $\mathcal{T}_k(G)$ is the union of $T_{k-1}(G)$ with the set of $\beta = \nabla\,(\beta', \beta'', next\,(\beta'))$ such that $\beta' \in T_{k-1}(G)$, $\beta'' \in T_{k-1}(G)$, $\beta'(next\,(\beta')) = \langle A, V_L, \mathsf{true}\rangle$ for some $A \in V_N$ and when $next\,(\beta') \notin spine\,(\beta')$ then $\beta'' \in aux\,(V_N, V_T, V_L, A)$ where β'' has no OA nodes. Otherwise (when $next\,(\beta') \in spine\,(\beta')$), $\beta'' \in \mathcal{A}_2 \cap aux\,(V_N, V_T, V_L, A)$.

It should be clear that $T(G)$ as defined in Section 2.2 is equal to the following set.

$$\{\, \nabla\,(\alpha, \beta, \epsilon) \mid \text{for some } k \geq 0\ \beta \in T_k(G) \cap aux\,(V_N, V_T, V_L, S) \text{ and } \beta \text{ has no OA nodes} \,\}$$

$L(G) = L(G')$ follows from Lemma 3.6. For convenience we first define the middle of a tree. For $\beta \in T_k(G)$ $(k \geq 0)$ let $middle\,(\beta) = \epsilon$ if β contains no OA nodes, otherwise $middle\,(\beta) = d$ where $d \in spine\,(\beta)$ and $|d| = |next\,(\beta)|$. In other words, the middle of a tree is the address of the node that is the closest node on the spine to the root that is as deep as any OA node in the tree. Note that for each tree $\beta \in T_k(G)$ for some $k > 0$ the entire terminal yield of the tree is dominated by the node with address $middle\,(\beta)$, i.e., $yield\,(\beta) = yield\,(\beta/middle\,(\beta))$. Note also that $\langle \beta, middle\,(\beta)\rangle$ is pruned.

Lemma 3.6 The following two statements are equivalent:

1. For all $A \in V_N$, $c \in cat\,(V_N')$, and $w_1, w_2 \in V_T^*$, there exists $\beta \in T_k(G) \cap aux\,(V_N, V_T, V_L, A)$, $k \geq 0$, such that $yield\,(\beta) = w_1 A w_2$, and $enc\,(\beta, middle\,(\beta)) = c$
2. there exist $c_1, \ldots, c_n$ and $u_1, \ldots, u_n$ such that $c \overset{*}{\underset{G'}{\Longrightarrow}} c_1 \ldots c_p \ldots c_n$ where c_p is the primary descendent of c, $target\,(c) = A$, $w_1 = u_1 \ldots u_p$, $w_2 = u_{p+1} \ldots u_n$, $u_j \in V_T^\epsilon$ and $c_j \in f(u_j)$ $(1 \leq j \leq n)$.

Proof We first prove that statement 1 implies statement 2 by induction on k. The case for $k = 0$ follows directly from the construction. Suppose that for some $k \geq 1$ $\beta \in T_k(G) \cap aux\,(V_N, V_T, V_L, A)$, $yield\,(\beta) = w_1 A w_2$, $\langle \beta, middle\,(\beta)\rangle$ is pruned and $enc\,(\beta, middle\,(\beta)) = c$. Let $\beta = \nabla\,(\beta', \beta'', next\,(\beta'))$ for $\beta', \beta'' \in T_{k-1}(G)$ and $\beta'(next\,(\beta')) = \langle B, V_L, \mathsf{true}\rangle$ for some $B \in V_N$.

- Suppose that $next\,(\beta') \notin spine\,(\beta')$ in which case $\beta'' \in aux\,(V_N, V_T, V_L, B)$ where β'' has no OA nodes. It is either the case that $next\,(\beta') = d1$ for some d (see case 1a of Figure 10) or $next\,(\beta') = d2$ for some d (see case 1b of Figure 10). These two cases are very similar so we only consider the first possibility in which the adjunction node is to the left of the spine.

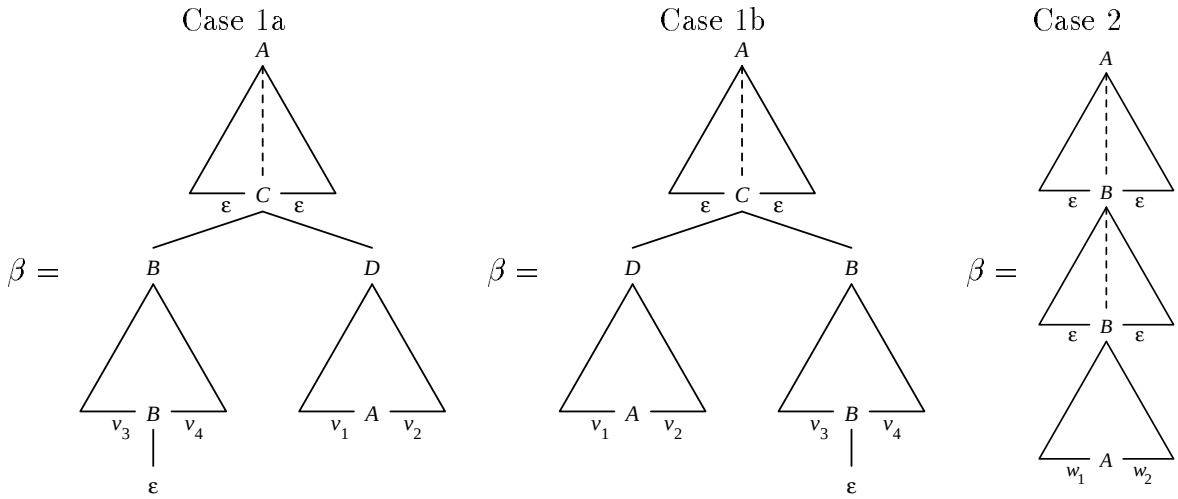

Figure 10: Three possibilities

In this case $d_2 = \mathit{middle}\,(\beta')$ and $\mathit{enc}\,(\beta', d2) = (c\backslash B)$. There are strings v_1, v_2, v_3 and v_4 such that $\mathit{yield}\,(\beta') = v_1 A v_2$ and $\mathit{yield}\,(\beta'') = v_3 B v_4$, thus, $w_1 = v_3 v_4 v_1$ and $w_2 = v_2$. By induction $(c\backslash B) \overset{*}{\underset{G}{\Longrightarrow}} c_1 \ldots c_p \ldots c_n$ where c_p is the primary descendent of $(c\backslash B)$, $\mathit{target}\,((c\backslash B)) = A$, $v_1 = u_1 \ldots u_p$, $v_2 = u_{p+1} \ldots u_n$ and $c_j \in f(u_j)$ $(1 \leq j \leq n)$. Since β'' has no OA nodes, $\mathit{middle}\,(\beta'') = \epsilon$ and $\mathit{enc}\,(\beta'', \epsilon) = B$. By induction we know that $B \overset{*}{\underset{G'}{\Longrightarrow}} c'_1 \ldots c'_m$, $v_3 v_4 = u'_1 \ldots u'_m$ and $c'_j \in f(u'_j)$ $(1 \leq j \leq m)$. We are not concerned with the primary descendent of B in this derivation. By the rules in R we have the following derivation.

$$c \underset{G'}{\Longrightarrow} B\,(c\backslash B) \overset{*}{\underset{G'}{\Longrightarrow}} c'_1 \ldots c'_m c_1 \ldots c_p \ldots c_n$$

where c_p is the primary descendent of c.

- Alternatively, suppose that $\mathit{next}\,(\beta') \in \mathit{spine}\,(\beta')$ (see case 2 of Figure 10) in which case $\beta'' \in \mathcal{A}_2 \cap \mathit{aux}\,(V_N, V_T, V_L, B)$ and, therefore, β'' contains no terminal symbols and $\mathit{yield}\,(\beta') = w_1 A w_2$. Note that $\mathit{middle}\,(\beta') = \mathit{next}\,(\beta')$. By the construction we know that $c'' \in f(\epsilon)$ where

$$\mathit{enc}\,(\beta'', \mathit{middle}\,(\beta'')) = \mathit{enc}\,(\beta'', \mathit{ft}\,(\beta'')) = c'' = (\ldots(B|_1 B_1)|_2 \ldots |_m B_m)$$

for some $B_i \in V'_N$ and some $m > 0$. In addition, we know that $\langle \beta, \mathit{next}\,(\beta')\,\mathit{middle}\,(\beta'')\rangle$ is pruned and that $\mathit{enc}\,(\beta', \mathit{next}\,(\beta')) = (c'/\hat{B})$ and

$$\mathit{enc}\,(\beta, \mathit{middle}\,(\beta)) = \mathit{enc}\,(\beta, \mathit{next}\,(\beta')\,\mathit{middle}\,(\beta'')) = (\ldots(c'|_1 B_1)|_2 \ldots |_m B_m) = c$$

By induction, $(c'/\hat{B}) \overset{*}{\underset{G'}{\Longrightarrow}} c_1 \ldots c_p \ldots c_n$ where c_p is the primary descendent of $(c'/\hat{B})$, $\mathit{target}\left((c'/\hat{B})\right) = A$, $w_1 = u_1 \ldots u_p$, $w_2 = u_{p+1} \ldots u_n$, and $c_j \in f(u_j)$ $(1 \leq j \leq n)$. By the combinatory rules in R we have the derivation

$$\begin{aligned} c = (\ldots(c'|_1 B_1)|_2 \ldots |_m B_m) &\underset{G'}{\Longrightarrow} (c'/\hat{B})\,(\ldots(\hat{B}|_1 B_1)|_2 \ldots |_m B_m) \\ &\overset{*}{\underset{G'}{\Longrightarrow}} c_1 \ldots c_p \ldots c_n (\ldots(\hat{B}|_1 B_1)|_2 \ldots |_m B_m) \end{aligned}$$

where c_p is the primary descendent of $(\ldots(c'|_1 B_1)|_2 \ldots |_m B_m)$ and since

$$(\ldots(\hat{B}|_1 B_1)|_2 \ldots |_m B_m) \in f(\epsilon)$$

we have the desired terminal strings w_1 and w_2 (take $u_{n+1} = \epsilon$).

We prove that statement 2 implies statement 1 of the lemma by induction on the number of derivation steps that are used in derivations in G'. The basis involves consideration of categories in the range of f. It follows from the construction that the lemma will hold of these categories.

Suppose that $c \overset{k}{\underset{G'}{\Longrightarrow}} c_1 \ldots c_p \ldots c_n$ where c_p is the primary descendent of c, $target\,(c) = B$, $w_1 = u_1 \ldots u_p$, $w_2 = u_{p+1} \ldots u_n$, $u_j \in V_T^\epsilon$ and $c_j \in f(u_j)$ $(1 \leq j \leq n)$.

- Suppose the combinatory rule $(x/B)\ B \rightarrow x$ is used in the first step of the derivation. There are l and p' where $p \leq l < p' \leq n$ such that

$$\begin{aligned} c &\underset{G'}{\Longrightarrow} (c/B)\ B \\ &\overset{k'}{\underset{G'}{\Longrightarrow}} c_1 \ldots c_p \ldots c_l\ B \\ &\overset{k''}{\underset{G'}{\Longrightarrow}} c_1 \ldots c_p \ldots c_l c_{l+1} \ldots c_{p'} \ldots c_n \end{aligned}$$

  where k' and k'' are less than k. Thus, by induction we have for some $k_1 \geq 0$ a tree $\beta_1 \in T_{k_1}(G) \cap aux\,(V_N, V_T, V_L, A)$ such that $target\,(c) = A$, $yield\,(\beta_1) = u_1 \ldots u_p A u_{p+1} \ldots u_l$, $\langle \beta_1, middle\,(\beta_1)\rangle$ is pruned and $enc\,(\beta_1, middle\,(\beta_1)) = (c/B)$. There is some $d \in dom\,(\beta_1)$ such that $middle\,(\beta_1) = d1$, $next\,(\beta_1) = d2$ and $\beta_1(d2) = \langle B, V_L, \mathsf{true}\rangle$.

  Also by induction we have for some $k_2 \geq 0$ a tree $\beta_2 \in T_{k_2}(G) \cap aux\,(V_N, V_T, V_L, B)$ such that $\langle \beta_2, middle\,(\beta_2)\rangle$ is pruned, $enc\,(\beta_2, middle\,(\beta_2)) = B$ (which implies that β_2 has no OA nodes) and

$$yield\,(\beta_2) = u_{l+1} \ldots u_{p'} B u_{p'+1} \ldots u_n$$

  Thus, $\beta = \nabla\,(\beta_1, \beta_2, d2) \in T_k(G)$ for some k where

$$yield\,(\beta) = u_1 \ldots u_p A u_{p+1} \ldots u_{p'} \epsilon u_{p'+1} \ldots u_n$$

  Furthermore, $\langle \beta, middle\,(\beta)\rangle$ is pruned, and $enc\,(\beta, middle\,(\beta)) = c$.

- The case in which the combinatory rule $B\ (x\backslash B) \rightarrow x$ is used is similar to the case just considered.

- Suppose that for some $0 \leq m \leq arity\,(c)$ the rule

$$(x/\hat{B})\ (\ldots(\hat{B}|_1 z_1)|_2 \ldots |_m z_m) \rightarrow (\ldots(x|_1 z_1)|_2 \ldots |_m z_m)$$

  is used in the first step of the derivation and that $c = (\ldots(c'|_1 c'_1)|_2 \ldots |_m c'_m)$. Thus we have

$$c = (\ldots(c'|_1 c'_1)|_2 \ldots |_m c'_m) \underset{G'}{\Longrightarrow} (c'/\hat{B})\ (\ldots(\hat{B}|_1 c'_1)|_2 \ldots |_m c'_m)$$

  Since a category whose target category is $\hat{B}$ cannot instantiate the right hand side of a rule in R and from the construction of f, it must be the case that

$$(\ldots(\hat{B}|_1 c'_1)|_2 \ldots |_m c'_m) \in f(\epsilon)$$

Thus $c_n = (\ldots(\hat{B}|_1 c'_1)|_2 \ldots|_m c'_m)$ and $u_n = \epsilon$ and

$$\begin{aligned} c = (\ldots(c'|_1 c'_1)|_2 \ldots|_m c'_m) &\underset{G'}{\Longrightarrow} (c'/\hat{B})\ (\ldots(\hat{B}|_1 c'_1)|_2 \ldots|_m c'_m) \\ &\overset{k'}{\underset{G'}{\Longrightarrow}} c_1 \ldots c_p \ldots c_{n-1}\ (\ldots(\hat{B}|_1 c'_1)|_2 \ldots|_m c'_m) \end{aligned}$$

where k' is less than k. Thus, by induction we have for some $k \geq 0$ a tree $\beta_1 \in T_k(G) \cap aux\,(V_N, V_T, V_L, A)$ such that $target\,(c') = A$, $yield\,(\beta_1) = u_1 \ldots u_p A u_{p+1} \ldots u_{n-1}$, $\langle \beta_1, middle\,(\beta_1) \rangle$ is a pruned tree and $enc\,(\beta_1, middle\,(\beta_1)) = (c'/\hat{B})$. We also have $next\,(\beta_1) = middle\,(\beta_1)$, (i.e., the next point of adjunction into β_1 will be on the spine) and $\beta_1(middle\,(\beta_1)) = \langle B, V_L, \mathsf{true} \rangle$.

Since $(\ldots(\hat{B}|_1 c'_1)|_2 \ldots|_m c'_m) \in f(\epsilon)$ there must be a tree $\beta_2 \in \mathcal{A}_2 \cap aux\,(V_N, V_T, V_L, B)$ such that $enc\,(\beta_2, ft\,(\beta_2)) = (\ldots(B|_1 c'_1)|_2 \ldots|_m c'_m)$. Note $yield\,(\beta_2) = B$.

Thus, $\beta = \nabla\,(\beta_1, \beta_2, middle\,(\beta_1)) \in T_{k+1}(G)$ where $yield\,(\beta) = u_1 \ldots u_p A u_{p+1} \ldots u_{n-1}$, $\langle \beta, middle\,(\beta) \rangle$ is a pruned tree, and $enc\,(\beta, middle\,(\beta)) = c = (\ldots(c'|_1 c'_1)|_2 \ldots|_m c'_m)$.

∎

Example 3.3 The tag

$$G = (\{\, S, A, B \,\}, \{\, a, b \,\}, \left\{ \overline{\beta_1}, \ldots, \overline{\beta_4} \right\}, S, \{\, \alpha \,\}, \mathcal{A}_1 \cup \mathcal{A}_2, _)$$

generates the language $\{\, a^n b^n \mid n \geq 0 \,\}$ where α is as shown in Figure 5, $\mathcal{A}_1$ contains the trees β_1 and β_2 and $\mathcal{A}_2$ contains the trees β_3 and β_4 all shown in Figure 11.

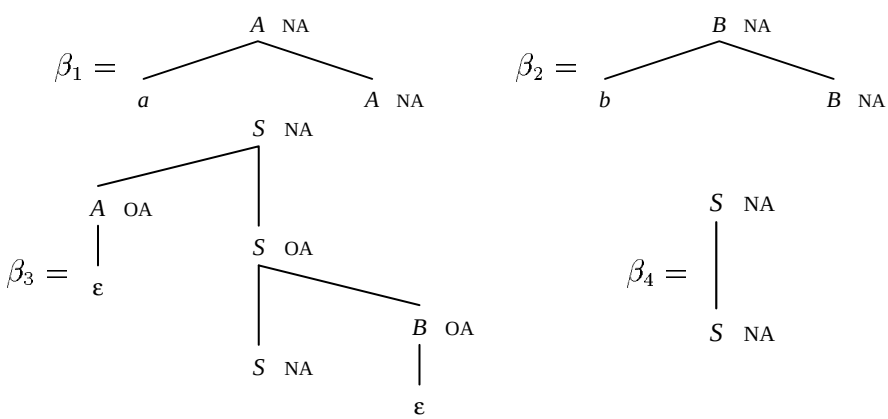

Figure 11: Auxiliary trees of G

The above construction generates the ccg

$$G' = (\left\{ S, \hat{S}, A, \hat{A}, B, \hat{B} \right\}, \{\, a, b \,\}, S, f, R)$$

where f is defined as follows.

$$f(a) = \{\, A \,\} \quad f(b) = \{\, B \,\} \quad f(\epsilon) = \left\{ (((S\backslash A)/\hat{S})/B), (((\hat{S}\backslash A)/\hat{S})/B), S, \hat{S} \right\}$$

and R includes the following rules (we have only included those that can be used in derivations of G'.

$$R = \left\{ \begin{array}{c} (x/B)\ B \rightarrow x, \quad A\ (x\backslash A) \rightarrow x, \quad (x/S)\ S \rightarrow x, \\ (x/\hat{S})\ (((\hat{S}\backslash A)/\hat{S})/B) \rightarrow (((x\backslash A)/\hat{S})/B), \quad (x/\hat{S})\ \hat{S} \rightarrow x \end{array} \right\}$$

4 Conclusions

As mentioned in the introduction, notational differences between ccg, hg, lig and tag can be understood in terms of the way in which they extend cfg. On the one hand, hg maintain the context-freeness of cfg while adding the operation of wrapping. On the other hand, lig and ccg make use of unbounded stack-like structures to control the use of rules in derivations. From the equivalence results presented in this paper, we conclude that these two, superficially very different, approaches are equivalent extensions of cfg. Since these are independently conceived formalisms, each intended to capture certain aspects of the structure of natural language, our result showing their equivalence lends credence to each of the approaches.

The understanding of the relationship between these two approaches gained from the constructions used in the proofs given in Section 3 has led to interesting discoveries regarding the relationship between parsing algorithms for these systems. Differences between these formalisms gave rise to what appear to be distinct styles of parsing algorithms. The CKY algorithm [9, 29] for cfg has been extended in two ways. In [15] a hg CKY-style parsing algorithm is given. This algorithm resembles the cfg case in that subsets of the nonterminal alphabet are stored in array entries. The extension over cfg involves the use of a four dimensional array to encode pairs of substrings of the input string. Given the close relationship between hg and tag established in this paper, it was possible to adapt this algorithm to give a tag parser [22].

In [23, 24] a lig CKY-style parsing algorithm is given. This algorithm extends the cfg case in that encodings of stacks are stored in the array entries. Given the close relationship between lig, ccg and tag described in this paper, it was possible to adapt this algorithm to give ccg and tag parsers [24].

Obviously, as a result of the weak equivalence of ccg, hg, lig and tag any result shown for one formalism applies to the other three. Such results include the definition of a string automaton for this class [22], various closure and decidability properties including the result that it forms a AFL [22, 17], and the definition of an infinite language hierarchy extending the progression from cfl to this class [26, 27].

References

[1] A. V. Aho. Indexed grammars — An extension to context free grammars. *J. ACM*, 15:647–671, 1968.

[2] K. Ajdukiewicz. Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27, 1935. English translation in: Polish logic 1920-1939, ed. by Storrs McCall, 207-231. Oxford University Press.

[3] C. Culy. The complexity of the vocabulary of bambara. *Ling. and Philosophy*, 8:345–351, 1985.

[4] J. Duske and R. Parchmann. Linear indexed languages. *Theor. Comput. Sci.*, 32:47–60, 1984.

[5] G. Gazdar. Applicability of indexed grammars to natural languages. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 69—94. D. Reidel, Dordrecht, Holland, 1988.

[6] G. Gazdar, E. Klein, G. K. Pullum, and I. A. Sag. *Generalized Phrase Structure Grammars.* Blackwell Publishing, Oxford, 1985. Also published by Harvard University Press, Cambridge, MA.

[7] A. K. Joshi. How much context-sensitivity is necessary for characterizing structural descriptions — Tree Adjoining Grammars. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Processing — Theoretical, Computational and Psychological Perspective*, pages 206—250. Cambridge University Press, New York, NY, 1985. Originally presented in 1983.

[8] A. K. Joshi, L. S. Levy, and M. Takahashi. Tree adjunct grammars. *J. Comput. Syst. Sci.*, 10(1):136—163, 1975.

[9] T. Kasami. An efficient recognition and syntax algorithm for context-free languages. Technical Report AF-CRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA, 1965.

[10] A. S. Kroch. Asymmetries in long distance extraction in a tag grammar. In M. Baltin and A. S. Kroch, editors, *New Conceptions of Phrase Structure.* University of Chicago, Press, Chicago, IL, 1986.

[11] A. S. Kroch. Subjacency in a tree adjoining grammar. In A. Manaster-Ramer, editor, *Mathematics of Language.* John Benjamins, Amsterdam, 1987.

[12] A. S. Kroch and A. K. Joshi. Linguistic relevance of tree adjoining grammars. Technical Report MS-CIS-85-18, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, 1985.

[13] A. S. Kroch and A. K. Joshi. Analyzing extraposition in a tree adjoining grammar. In G. Huck and A. Ojeda, editors, *Syntax and Semantics: Discontinuous Constituents.* Academic Press, New York, NY, 1986.

[14] A. S. Kroch and B. Santorini. The derived constituent structure of West Germanic verb-raising construction. In *Proceedings of the Princeton Workshop on Grammar*, 1987.

[15] C. Pollard. *Generalized Phrase Structure Grammars, Head Grammars and Natural Language.* PhD thesis, Stanford University, 1984.

[16] G. Pullum and G. Gazdar. Natural languages and context-free languages. *Ling. and Philosophy*, 4:471–504, 1982.

[17] K. Roach. Formal properties of head grammars. In A. Manaster-Ramer, editor, *Mathematics of Language.* John Benjamins, Amsterdam, 1987.

[18] B. Santorini. The West Germanic verb-raising construction: A tree adjoining grammar analysis. Master's thesis, University of Pennsylvania, Philadelphia, PA, 1986.

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

[20] M. Steedman. Combinators and grammars. In R. Oehrle, E. Bach, and D. Wheeler, editors, *Categorial Grammars and Natural Language Structures*, pages 417—442. Foris, Dordrecht, 1986.

[21] M. J. Steedman. Dependency and coordination in the grammar of Dutch and English. *Language*, 61:523–568, 1985.

[22] K. Vijay-Shanker. *A Study of Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1987.

[23] K. Vijay-Shanker and D. J. Weir. The recognition of combinatory categorial grammars and linear indexed grammars. In M. Tomita, editor, *Current Issues in Parsing Technology*. Kluwer Academic Publishers, 1991.

[24] K. Vijay-Shanker and D. J. Weir. Parsing constrained grammar formalisms. *Comput. Ling.*, in press.

[25] K. Vijay-Shanker, D. J. Weir, and A. K. Joshi. Tree adjoining and head wrapping. In 11th *International Conference on Comput. Ling.*, pages 202–207, 1986.

[26] D. J. Weir. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1988.

[27] D. J. Weir. Automata theory as relevant to linguistics. In W. Bright, editor, *Oxford International Encyclopedia of Linguistics*, volume 1, pages 145—146. Oxford University Press, Oxford, England, 1992.

[28] D. J. Weir and A. K. Joshi. Combinatory categorial grammars: Generative power and relationship to linear context-free rewriting systems. In 26th *meeting Assoc. Comput. Ling.*, pages 278—285, 1988.

[29] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Inf. Control*, 10(2):189–208, 1967.

A A normal form for tag

We give a five phase conversion process to show that for every tag there is an equivalent tag that satisfies the conditions given in Section 3.4. We do not give a detailed proof since the conversion at each stage is straightforward.

1. Let $G_1 = (V_N, V_T, V_{L,1}, S, \mathcal{I}_1, \mathcal{A}_1, \text{-})$ be a tag. We assume that trees in $\mathcal{I}_1 \cup \mathcal{A}_1$ do not contain any OA nodes that are also NA nodes. We also assume that for every node label $\langle A, \mathsf{sa}, \mathsf{oa}\rangle$: $\mathsf{sa} \subseteq \left\{ \overline{\beta} \mid \beta \in aux\,(V_N, V_T, V_{L,1}, A) \right\}$. G_1 can be converted into an equivalent tag,
$$G_2 = (V_N \cup \{ S' \}, V_T, V_{L,2}, S', \{ \alpha \}, \mathcal{A}_1 \cup \mathcal{A}_2, \text{-})$$
where α is such that $\alpha(\epsilon) = \langle S', \mathsf{sa}_{S'}, \mathsf{true}\rangle$ and $\alpha(1) = \epsilon$ where $\mathsf{sa}_{S'} = \left\{ \overline{\beta} \mid \beta \in \mathcal{A}_2 \right\}$. For each tree $\alpha' \in \mathcal{I}_1$ a tree $\beta_{\alpha'}$ is included in $\mathcal{A}_2$ where $\beta_{\alpha'}/1 = \alpha'$, $\beta_{\alpha'}(\epsilon) = \beta_{\alpha'}(2) = \langle S', \emptyset, \mathsf{false}\rangle$ for some new nonterminal S'. The left subtree of $\beta_{\alpha'}$ is α' and the right subtree is a single node that is the foot of the tree. Note that the tree $\nabla(\alpha, \beta_{\alpha'}, \epsilon)$ corresponds very closely to α' as shown in Figure 12.

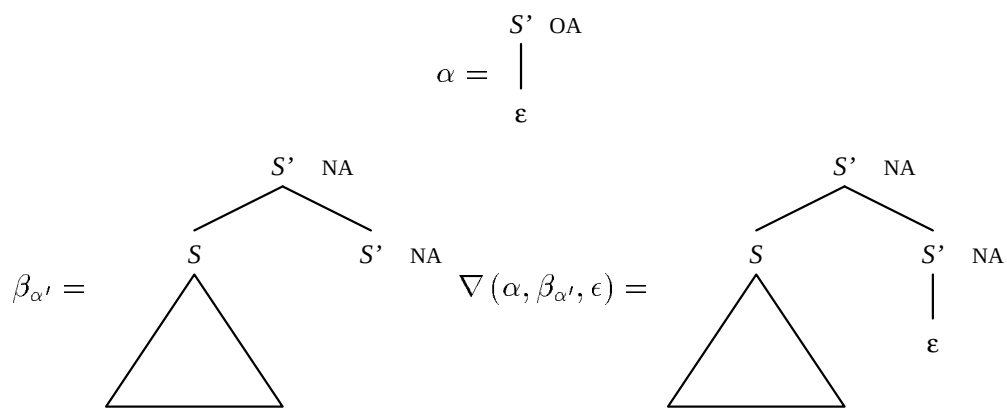

Figure 12: Conversion of initial trees

$V_{L,2} = V_{L,1} \cup \left\{ \overline{\beta} \mid \beta \in \mathcal{A}_2 \right\}$. Technically, we should specify different tree labelling function for each grammar. However, for each grammar the function denoted by - should be clear from the set of tree labels of the grammar.

2. From G_2 we construct an equivalent tag, $G_3 = (V_N \cup \{ S' \}, V_T, V_{L,3}, S', \{ \alpha \}, \mathcal{A}_3, \text{-})$, such that every node of a tree in $\mathcal{A}_3$ that is labelled by a nonterminal either has a NA constraint or an OA constraint. In other words, there will be no nodes at which adjunction is optional. For each $\beta \in \mathcal{A}_1 \cup \mathcal{A}_2$ and $D \subseteq dom\,(\beta)$ such that for all $d \in D$ we have $\beta(d) = \langle A, \mathsf{sa}, \mathsf{false}\rangle$ for some $A \in V_N$ and nonempty $\mathsf{sa} \subseteq V_{L,2}$ include β_D in $\mathcal{A}_3$. For each $d \in \mathcal{N}_+^*$
$$\beta_D(d) = \begin{cases} \langle A, \mathsf{sa}, \mathsf{true}\rangle & \text{if } d \in D \text{ and } \beta(d) = \langle A, \mathsf{sa}, \mathsf{false}\rangle \\ \langle A, \emptyset, \mathsf{false}\rangle & \text{if } d \notin D \text{ and } \beta(d) = \langle A, \mathsf{sa}, \mathsf{false}\rangle \text{ for some } \mathsf{sa} \\ \beta(d) & \text{otherwise} \end{cases}$$
$V_{L,3} = \left\{ \overline{\beta} \mid \beta \in \mathcal{A}_3 \right\}$.

3. From G_3 we construct an equivalent tag, $G_4 = (2^{V_{L,3}}, V_T, V_{L,4}, \mathsf{sa}_{S'}, \{ \alpha' \}, \mathcal{A}_4, \text{-})$, such that the root and foot of each tree in $\mathcal{A}_4$ has an NA constraint and all other internal

nodes have OA constraints with no restriction on which trees can be adjoined except that the nonterminals must match. To do this we use the set of labels forming the SA constraint at each node as its nonterminal label. We want a tree β to be adjoinable at a node whose nonterminal label contains $\overline{\beta}$. Thus, for every set of labels sa containing $\overline{\beta}$ there must be an instance of β with a new root and foot nodes having nonterminal label sa.

α' is such that $\alpha'(1) = \epsilon$ and $\alpha'(\epsilon) = \langle \mathsf{sa}_{S'}, V_{L,4}, \mathsf{true} \rangle$ where $\mathsf{sa}_{S'}$ is such that $\alpha(\epsilon) = \langle S', \mathsf{sa}_{S'}, \mathsf{true} \rangle$.

For each $\beta \in \mathcal{A}_3$ let β' be such that $\beta'(d) = \langle \mathsf{sa}, V_{L,4}, \mathsf{true} \rangle$ if $\beta(d) = \langle A, \mathsf{sa}, \mathsf{oa} \rangle$ for some A and oa, and $\beta'(d) = \beta(d)$ otherwise. For each $\mathsf{sa} \subseteq V_{L,3}$ such that $\overline{\beta} \in \mathsf{sa}$ include the tree β_{sa} in $\mathcal{A}_4$ where $\beta_{\mathsf{sa}}/1 = \beta'$, $\beta_{\mathsf{sa}}(\epsilon) = \beta_{\mathsf{sa}}(ft\,(\beta)\,1) = \langle \mathsf{sa}, \emptyset, \mathsf{false} \rangle$.

Also include in $\mathcal{A}_4$ a tree $\beta_\emptyset$ where $\beta_\emptyset(\epsilon) = \beta_\emptyset(1) = \langle \emptyset, \emptyset, \mathsf{false} \rangle$ and $ft\,(\beta_\emptyset) = 1$.

$V_{L,4} = \left\{ \overline{\beta} \mid \beta \in \mathcal{A}_4 \right\}$.

4. From G_4 we describe an equivalent tag $G_5 = (V'_N, V_T, V_{L,5}, \mathsf{sa}_{S'}, \{\, \alpha' \,\}, prune\,(\mathcal{A}_4), \text{–})$, where

$$prune\,(\mathcal{A}) = \bigcup_{\beta \in \mathcal{A}} prune\,(\beta)$$

and $prune\,(\beta)$ is defined below. First, note that tree substitution can be simulated (with respect to terminal yield) by adjunction if the node at which adjunction takes place dominates a single node labelled by ϵ. Informally, a tree is pruned by removing subtrees rooted at siblings of the spine and arranging that a simulated substitution of the removed subtree can occur. The removed subtree will be turned into an auxiliary tree with the addition of a foot node and itself be pruned. Thus, through a series of simulated substitutions a tree corresponding closely to the original tree can be recreated.

Let $\beta' \in prune\,(\beta)$ where for each $d \in \mathcal{N}^*_+$

$$\beta'(d) = \begin{cases} \beta(d) & \text{if } d \in spine\,(\beta) \\ \langle (\beta, d), V_{L,5}, \mathsf{true} \rangle & \text{if } d \notin spine\,(\beta) \text{ and } d = d'i \text{ where} \\ & i \geq 1 \text{ and } d' \in spine\,(\beta) \\ \epsilon & \text{if } d \notin spine\,(\beta) \text{ and } d = d'i1 \text{ where} \\ & i \geq 1,\ d' \in spine\,(\beta), \text{ and } d'i \notin spine\,(\beta) \\ \text{undefined} & \text{otherwise} \end{cases}$$

β' is produced from β such that every node with address d that is the sibling of a node on the path from the root to foot of β is relabelled with the new nonterminal (β, d), given an OA constraint, and, in addition, the subtree under this node is replaced by a single node labelled ϵ.

For each $d \in dom\,(\beta)$ such that $d \notin spine\,(\beta)$ and $d = d'i$ where $i \geq 1$ and $d' \in spine\,(\beta)$ (i.e., d is the sibling of a node on spine of β) trees are included in $prune\,(\beta)$ as follows.

If $\beta(d) = w$ for some $w \in V_T \cup \{\, \epsilon \,\}$ then include the tree β_w in $prune\,(\beta)$ where for each $d' \in \mathcal{N}^*_+$

$$\beta_w(d') = \begin{cases} \langle (\beta, d), \emptyset, \mathsf{false} \rangle & \text{if } d' = \epsilon \text{ or } d' = 2 \\ w & \text{if } d' = 1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

Otherwise, let $prune\,(\beta_d) \subseteq prune\,(\beta)$ where β_d is a slight modification of the subtree β/d that is defined as follows.

Let β'_d be such that $\beta'_d/1 = \beta/d$ and $\beta'_d(\epsilon) = \langle(\beta, d), \emptyset, \mathsf{false}\rangle$, i.e., β'_d is the subtree of β rooted at the node with address d with the addition of a new root with nonterminal label (β, d) and with an NA constraint. We now add a foot node to β'_d to get β_d. Let d' be such that $\beta'_d(d') = \langle A, \mathsf{sa}, \mathsf{oa}\rangle$ for some A, sa and oa and for all d'' such that $\beta_d(d'') = \langle A, \mathsf{sa}, \mathsf{oa}\rangle$ for some A, sa and oa $|d'| \geq |d''|$ (d' is the address of a deepest nonterminal node in β'_d). Let $i > 1$ be such that $d'(i-1) \in dom\,(\beta'_d)$ and $d'i \notin dom\,(\beta'_d)$. Let β_d be equal to β'_d except that $\beta_d(d'i) = \langle(\beta, d), \emptyset, \mathsf{false}\rangle$.

$V_{L,5} = \left\{ \overline{\beta} \mid \beta \in prune\,(\mathcal{A}_4) \right\}$.

The tree, β, at the left of Figure 13 would be converted into the right four trees in the figure.

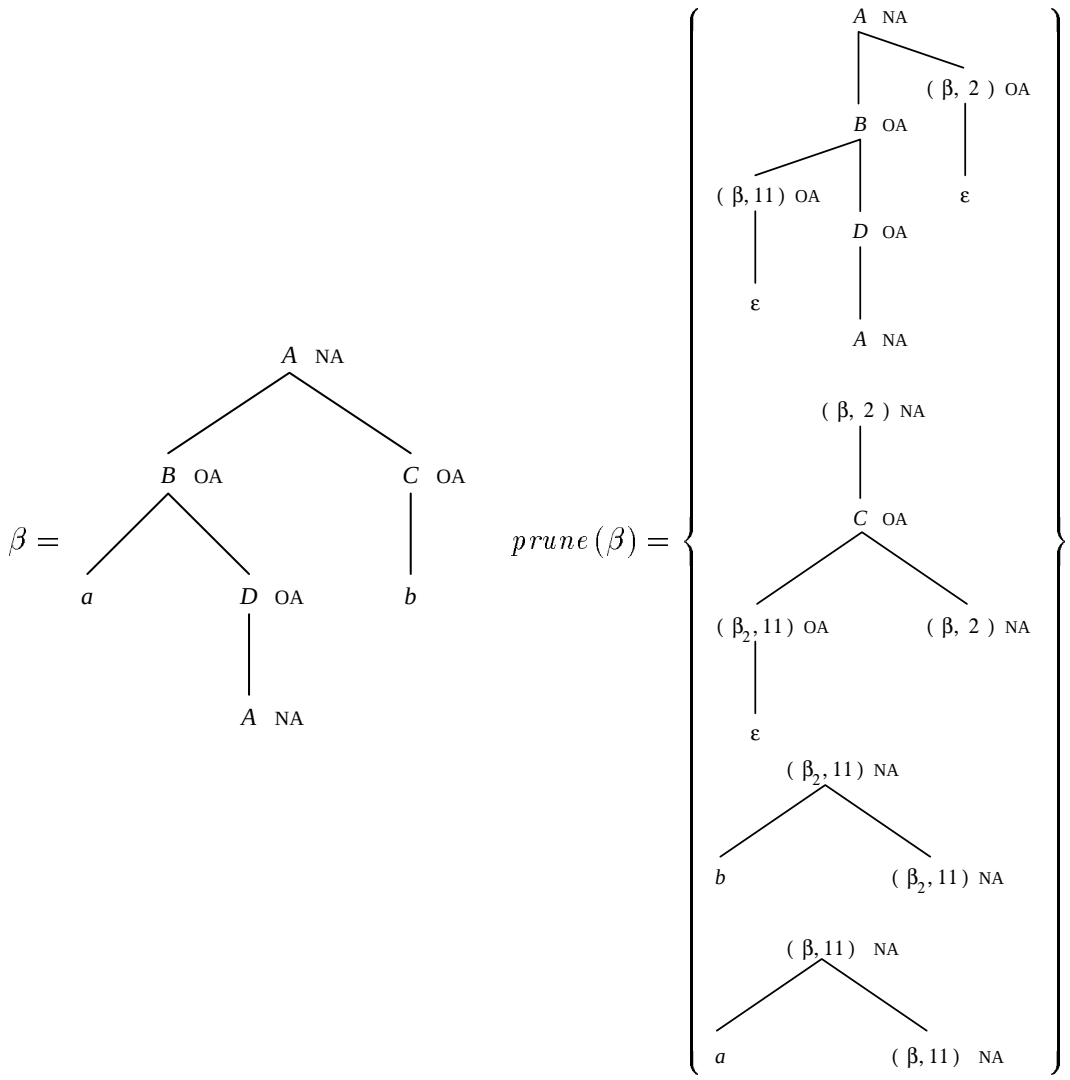

Figure 13: Pruned trees

5. Finally, we give a tag $G = (V'_N, V_T, V_{L,5}, \mathsf{sa}_{S'}, \{\,\alpha'\,\}, \mathcal{A}, _)$, equivalent to G_4, such that for every β in $\mathcal{A}$ there is at most one OA node at any level of β and the nodes of β have at most 2 children. Let β be a tree in $prune\,(\mathcal{A}_4)$. Note that from the above construction only nodes on the spine of β can have more than 1 child. The conversion of β is shown in Figure 14 in which we show how an arbitrary node on the spine of β can be stretched out to satisfy the above condition. Every node on the spine would be treated in the

same way. Note that where adjunction constraints remain unchanged they have been omitted.

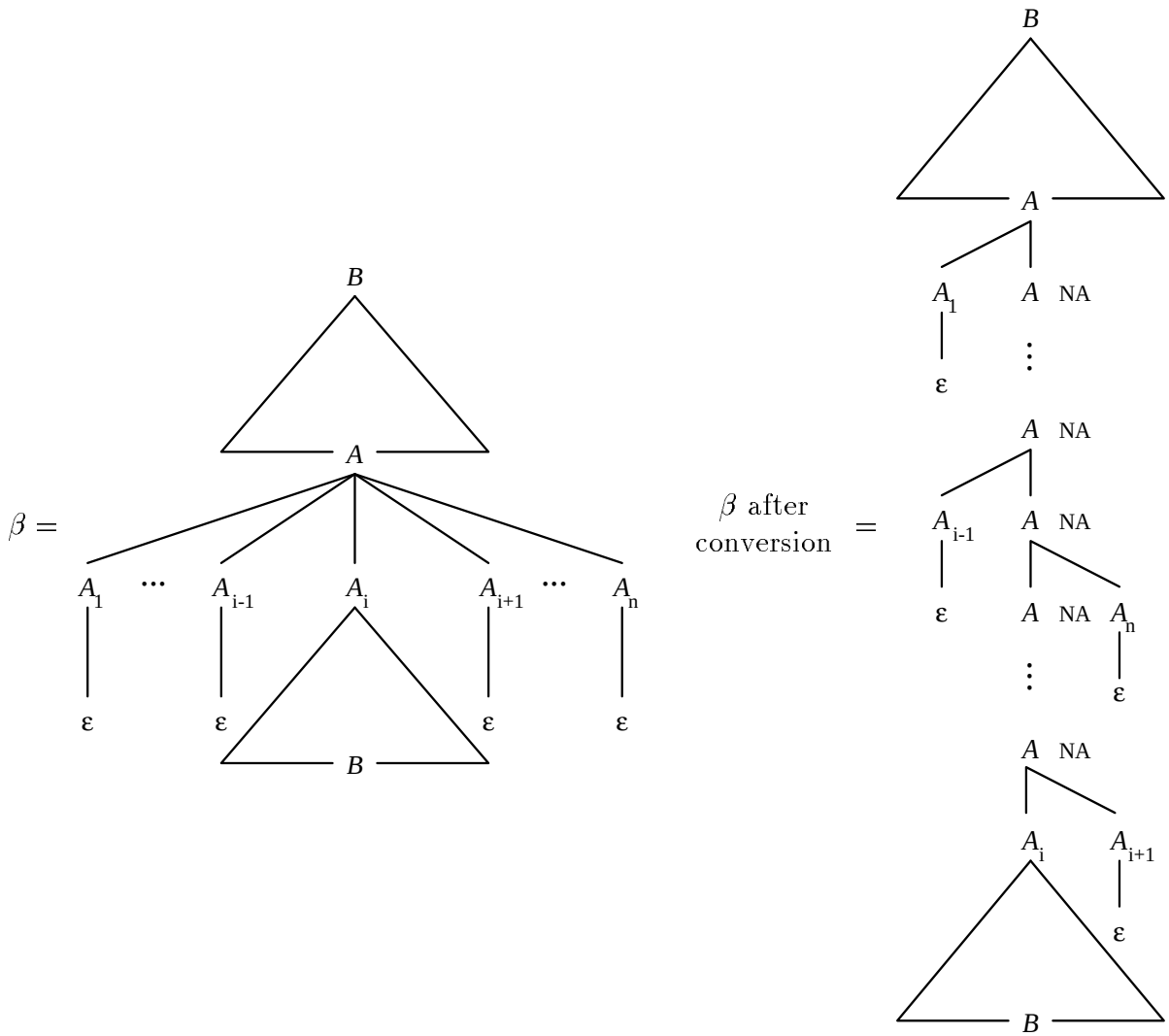

Figure 14: Conversion to binary branching