

# Language Theory and Infinite Graphs

Colin Stirling  
School of Informatics  
University of Edinburgh  
Edinburgh EH9 3JZ, UK  
email: cps@inf.ed.ac.uk

## Abstract

Automata and language theory study finitely presented mechanisms for generating languages. A language is a family of words. The Chomsky hierarchy of languages can be generated using grammars or using automata. At the lowest level are regular languages which are also generated by finite-state automata. At the next level are context-free languages. These are generated by context-free grammars and also by pushdown automata. Beyond this are the context-sensitive languages and the recursively enumerable languages, generated by linear bounded Turing machines and Turing machines.

A slight shift in focus is very revealing. Instead of grammars and automata as language generators, one views them as propagators of possibly infinite labelled transition graphs. This is our starting point. We shall examine various kinds of infinite graph, concentrating on pushdown automata and context-free grammars and also consider bisimulation equivalence as an alternative to language equivalence.

The main goal of the lectures is to provide decision procedures for infinite state graphs. We concentrate on proving decidability of language equivalence using both graph theoretic and combinatorial arguments. The main result to be examined is decidability of the DPDA equivalence problem: that language equivalence is decidable for deterministic context-free languages. Despite intensive work throughout the late 1960s and 1970s, this problem remained unsolved until 1997 when Sénizergues announced a positive solution. His proof consisted of two semi-decision procedures, and so there was no complexity bound on the procedure. In the lectures we provide a deterministic decision procedure with a primitive recursive upper bound.

We shall also briefly discuss the open problem whether language equivalence is decidable for deterministic higher-order pushdown automata.

# Contents

<b>1</b>	<b>Introducing Pushdown Automata</b>	<b>3</b>
1.1	Pushdown graphs . . . . .	3
1.2	Collapsed graphs . . . . .	5
1.3	Subclasses of pushdown automata . . . . .	6
1.4	Decision questions . . . . .	7
1.5	Bisimulation equivalence . . . . .	8
1.6	The DPDA equivalence problem . . . . .	9
<b>2</b>	<b>Decidability of Equivalence of Simple Grammars</b>	<b>11</b>
2.1	Pushdown grammars . . . . .	11
2.2	Simple grammars . . . . .	12
2.3	Tableau proof rules . . . . .	14
2.4	Successful tableaux . . . . .	17
2.5	Correctness of tableaux . . . . .	19
<b>3</b>	<b>A Deeper Normal Form for DPDA</b>	<b>22</b>
3.1	Congruence and cancellation . . . . .	22
3.2	Strict deterministic grammars . . . . .	23
3.3	Sum configurations . . . . .	26
3.4	Characterising DPDA . . . . .	28
<b>4</b>	<b>Decidability of DPDA Equivalence</b>	<b>30</b>
4.1	Heads, tails and extensions . . . . .	30
4.2	Dynamic properties of head and tail forms . . . . .	31
4.3	The tableau proof system . . . . .	33
4.4	Successful tableaux . . . . .	38
4.5	Exercises . . . . .	42
<b>5</b>	<b>Proofs</b>	<b>43</b>
5.1	Proof of the extension theorem . . . . .	43
5.2	Correctness of the decision procedure . . . . .	44
<b>6</b>	<b>Higher-Order Pushdown Automata</b>	<b>49</b>

# 1 Introducing Pushdown Automata

Pushdown automata and their graphs are now introduced.

## 1.1 Pushdown graphs

The following four sets are ingredients of a pushdown automaton (a PDA).

- A finite set of states  $P$
- A finite set of stack symbols  $S$
- A finite alphabet  $A$
- A finite set of basic transitions  $T$

A basic transition has the form  $pX \xrightarrow{a} q\alpha$  where  $p$  and  $q$  are states in  $P$ ,  $X$  is a stack symbol in  $S$ ,  $a \in A \cup \{\epsilon\}$  and  $\alpha$  is a sequence of stack symbols in  $S^*$ .

A *configuration* of a PDA consists of a state  $p \in P$  and a sequence of stack symbols  $\beta \in S^*$ , written  $p\beta$ . The transitions of a configuration are defined by the following prefix rule where  $\delta$  is any sequence of stack symbols.

$$\text{PRE} \quad \text{If } pX \xrightarrow{a} q\alpha \in T \text{ then } pX\delta \xrightarrow{a} q\alpha\delta$$

A traditional automaton interpretation is that on input  $a$  the configuration  $pX\delta$  in state  $p$  with  $X$  at the top of the stack changes to state  $q$  and  $\alpha$  replaces  $X$ . Alternatively, with respect to a generational or process calculus perspective the configuration  $pX\delta$  generates, or performs,  $a$  and becomes  $q\alpha\delta$ . In both these accounts  $\epsilon$ -transitions have a special status. If  $a = \epsilon$  then configuration  $pX\delta$  may change to  $q\alpha\delta$  without reading an input or  $pX\delta$  may become  $q\alpha\delta$  silently without performing an observable action.

For simplicity, the following disjointness condition on  $T$  is assumed.

$$\text{If } pX \xrightarrow{\epsilon} q\beta \in T \text{ and } pX \xrightarrow{a} r\lambda \in T \text{ then } a = \epsilon$$

A configuration is *unstable*, only has  $\epsilon$ -transitions, or *stable*, has no  $\epsilon$ -transitions.

The *transition graph*  $G(p\beta)$  is generated by deriving all possible transitions from  $p\beta$  and every configuration reachable from it using the rule PRE.

**Example 1** The first example involves ingredients which are all singleton sets.

$$\begin{aligned} P &= \{p\} & S &= \{X\} \\ A &= \{a\} & T &= \{pX \xrightarrow{a} pXX\} \end{aligned}$$

$G(pX)$  is:  $pX \xrightarrow{a} pXX \xrightarrow{a} pXXX \xrightarrow{a} \dots$ . The transition  $pXXX \xrightarrow{a} pXXXX$  follows by applying PRE to  $pX \xrightarrow{a} pXX \in T$  with  $\delta = XX$ .  $\square$

**Example 2** Let  $P = \{p\}$ ,  $S = \{X, A, B\}$  and  $A = \{a, b\}$ . The basic transitions are

$$\begin{array}{lll} pX \xrightarrow{a} pAX & pA \xrightarrow{a} pAA & pB \xrightarrow{a} pAB \\ pX \xrightarrow{b} pBX & pA \xrightarrow{b} pBA & pB \xrightarrow{b} pBB \end{array}$$

$G(pX)$  is depicted in figure 1. The graph  $G(pX)$  is the full binary tree.  $\square$

**Example 3** Let  $P = \{p, q, r\}$ ,  $S = \{X\}$ ,  $A = \{a, b\}$  and  $T$  be

$$\begin{array}{lll} pX \xrightarrow{a} pXX & pX \xrightarrow{b} r\epsilon & rX \xrightarrow{\epsilon} r\epsilon \\ pX \xrightarrow{b} q\epsilon & qX \xrightarrow{b} q\epsilon & \end{array}$$

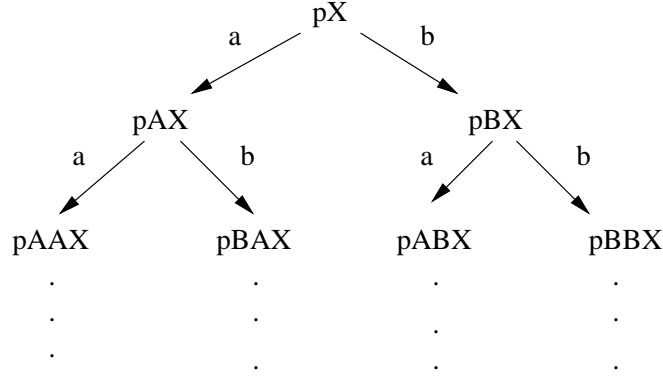


Figure 1:  $G(pX)$

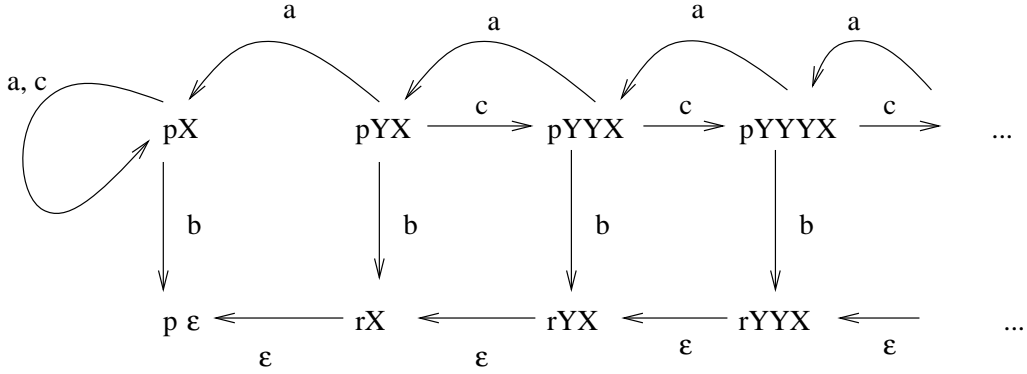


Figure 2: A deterministic PDA

The graph  $G(pX)$  is

$$\begin{array}{ccccccc}
 q\epsilon & \xleftarrow{b} & qX & \xleftarrow{b} & qXX & \xleftarrow{b} & \dots \\
 \uparrow b & & \uparrow b & & \uparrow b & & \\
 pX & \xrightarrow{a} & pXX & \xrightarrow{a} & pXXX & \xrightarrow{a} & \dots \\
 \downarrow b & & \downarrow b & & \downarrow b & & \\
 r\epsilon & \xleftarrow{\epsilon} & rX & \xleftarrow{\epsilon} & rXX & \xleftarrow{\epsilon} & \dots
 \end{array}$$

This example involves both  $\epsilon$ -transitions and *nondeterminism*: the configuration  $pXX^n$  has  $b$ -transitions to  $qX^n$  and  $rX^n$ . Note that  $\epsilon$  is used both as a transition label and for the empty stack sequence. Configurations  $q\epsilon$  and  $r\epsilon$  are terminal, as they do not have transitions.  $\square$

**Example 4** Often we don't spell out all the components of a PDA except its transitions. Figure 2 depicts a simple deterministic PDA whose basic transitions are

$$\begin{array}{cccc}
 pX \xrightarrow{a} pX & pX \xrightarrow{b} p\epsilon & pX \xrightarrow{c} pX & rX \xrightarrow{\epsilon} p\epsilon \\
 pY \xrightarrow{a} p\epsilon & pY \xrightarrow{b} r\epsilon & pY \xrightarrow{c} pYY & rY \xrightarrow{\epsilon} r\epsilon
 \end{array}$$

There is at most one transition  $pX \xrightarrow{a} q\alpha \in \mathbb{T}$  for  $p \in \mathbb{P}$ ,  $X \in \mathbb{S}$ ,  $a \in \mathbb{A} \cup \{\epsilon\}$ .  $\square$

A PDA is presentable in normal form, up to isomorphism of transition graphs, where each transition  $pX \xrightarrow{a} q\alpha \in \mathbb{T}$  obeys the constraint that the length of  $\alpha$ ,  $|\alpha|$ , is at most 2. Enforcement of the normal form is easy to achieve, by introducing extra stack symbols. Assume that the maximum length of a sequence of stack symbols

$\alpha$  in any transition  $pX \xrightarrow{a} q\alpha \in \mathsf{T}$  is  $n$ . New stack symbols  $[\beta]$  are introduced to achieve this normal form for sequences  $\beta$  of stack symbols whose length is at most  $n$ . The concrete construction is best illustrated by example.

**Example 5** Suppose the transitions  $\mathsf{T}$  are

$$\begin{array}{ll} pX \xrightarrow{a} pX_1X_2X_3X_4 & pX_1 \xrightarrow{a} p\epsilon \\ pX_2 \xrightarrow{a} pX_1X_3X_4 & pX_3 \xrightarrow{a} pX_1X_1X_1X_4 \\ pX_4 \xrightarrow{a} pX_1 & \end{array}$$

The largest size of a sequence of stack symbols introduced by a transition is 4. Therefore we only need to introduce extra stack elements  $[\beta]$  in the case that  $\beta$  has length at most 4. The transition  $pX \xrightarrow{a} pX_1[X_2X_3X_4]$  involving the new stack symbol  $[X_2X_3X_4]$  replaces the transition  $pX \xrightarrow{a} pX_1X_2X_3X_4$ . With respect to this initial transition, which obeys the normal form, further transitions and new stack symbols are introduced as follows.

$$\begin{array}{ll} p[X_2X_3X_4] \xrightarrow{a} p[X_1X_3X_4][X_3X_4] & p[X_1X_3X_4] \xrightarrow{a} p[X_3X_4] \\ p[X_3X_4] \xrightarrow{a} p[X_1X_1X_1X_4]X_4 & p[X_1X_1X_1X_4] \xrightarrow{a} p[X_1X_1X_4] \\ p[X_1X_1X_4] \xrightarrow{a} p[X_1X_4] & p[X_1X_4] \xrightarrow{a} pX_4 \end{array}$$

Transitions for  $pX_1$  and  $pX_4$  are as in  $\mathsf{T}$ . The reformulation of the PDA obeys the normal form. Clearly the two graphs  $\mathsf{G}(pX)$  with respect to the automaton and its normal form are isomorphic.  $\square$

A transition graph  $\mathsf{G}(p\alpha)$  has both bounded in and out degree. The out degree of a terminal configuration  $p\alpha$  is 0 and the out degree of a configuration  $pX\alpha$  is bounded by the number of basic transitions in  $\mathsf{T}$  which have the form  $pX \xrightarrow{a} q\beta$ . The in degree of a configuration  $p\alpha$  is bounded by the number of basic transitions in  $\mathsf{T}$  which have the form  $qY \xrightarrow{a} p\alpha'$ , when  $\alpha'$  is a prefix of  $\alpha$ .

## 1.2 Collapsed graphs

A natural extension of transitions is to words  $w \in \mathsf{A}^*$ . The extended transition  $p\alpha \xrightarrow{w} q\beta$  represents that there is a  $w$ -path from configuration  $p\alpha$  to configuration  $q\beta$  (and therefore  $p\alpha \xrightarrow{\epsilon} p\alpha$  for any configuration  $p\alpha$ ). The role of basic  $\epsilon$ -transitions thereby differs from basic  $a$ -transitions when  $a \in \mathsf{A}$ , because  $\epsilon$  is the empty word. For instance,  $pXX \xrightarrow{ab} r\epsilon$  in the case of example 3 of the previous section because  $pXX \xrightarrow{a} pXXX \xrightarrow{b} rXX \xrightarrow{\epsilon} rX \xrightarrow{\epsilon} r\epsilon$ .

A configuration of a PDA is either stable or unstable. To capture word transitions, the *collapsed* graph of a PDA, which abstracts from  $\epsilon$ -transitions, is defined. Consider just the stable configurations of example 3 of the previous section, and transitions  $\xrightarrow{a}$ ,  $a \in \mathsf{A}$ , between them. This is the collapsed graph, without  $\epsilon$ -transitions. For instance, its collapsed transition graph with stable root  $pX$ , written  $\mathsf{G}^c(pX)$ , is depicted in figure 3. In a collapsed graph unstable configurations are removed, and  $p\alpha \xrightarrow{a} q\beta$  if  $p\alpha \xrightarrow{a} r\gamma \xrightarrow{\epsilon} q\beta$ . If a PDA does not contain  $\epsilon$ -transitions then  $\mathsf{G}(p\alpha)$  is the same graph as  $\mathsf{G}^c(p\alpha)$ .

A collapsed transition graph may have infinite in or out degree. The configuration  $r\epsilon$  in figure 3 has infinite in degree because there are infinitely many transitions into it.

**Example 1** Assume  $\mathsf{T}$  is

$$rX \xrightarrow{a} pX \quad pX \xrightarrow{\epsilon} pXX \quad pX \xrightarrow{\epsilon} q\epsilon \quad qX \xrightarrow{b} q\epsilon$$

The graph  $\mathsf{G}(rX)$  is

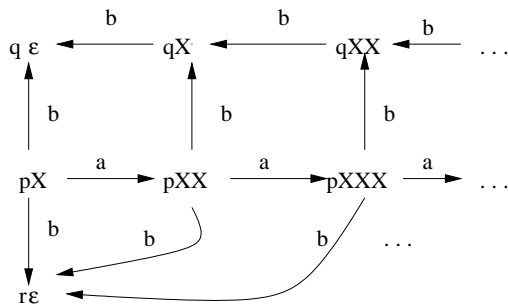


Figure 3: Collapsed graph  $G^c(pX)$

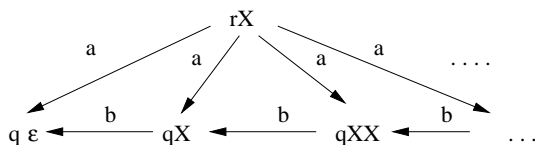


Figure 4: The collapsed graph  $G^c(rX)$

$$\begin{array}{ccccccc}
 rX & \xrightarrow{a} & pX & \xrightarrow{\epsilon} & pXX & \xrightarrow{\epsilon} & pXXX & \xrightarrow{\epsilon} & \dots \\
 & & \downarrow \epsilon & & \downarrow \epsilon & & \downarrow \epsilon & & \\
 & & q\epsilon & \xleftarrow{b} & qX & \xleftarrow{b} & qXX & \xleftarrow{b} & \dots
 \end{array}$$

The stable configurations in this graph are  $\{rX, qX^n : n \geq 0\}$ , and the collapsed graph  $G^c(rX)$  is presented in figure 4. Infinite out degree, as illustrated by this example, depends on nondeterminism in the basic  $\epsilon$ -transitions.  $\square$

What is the collapsed graph when there are not appropriate stable configurations? Consider a case involving just the following pair of basic transitions,  $pX \xrightarrow{a} qX$  and  $qX \xrightarrow{\epsilon} qXX$ . The graph  $G(pX)$  is

$$pX \xrightarrow{a} qX \xrightarrow{\epsilon} qXX \xrightarrow{\epsilon} qXXX \xrightarrow{\epsilon} \dots$$

The collapsed graph  $G^c(pX)$  is just the singleton node  $pX$ , because there are no stable configurations reachable from  $pX$ .

### 1.3 Subclasses of pushdown automata

There are many subclasses of pushdown automata. The following are important cases, some of which we shall examine later.

- Real-time: there are no  $\epsilon$ -transitions, so graphs  $G(p\alpha)$  and  $G^c(p\alpha)$  coincide.
- Single-state: the set of states  $P$  is a singleton set.
- $\epsilon$ -deterministic: if  $pX \xrightarrow{\epsilon} q\beta$ ,  $pX \xrightarrow{\epsilon} r\gamma \in \mathbb{T}$  then  $q = r$  and  $\beta = \gamma$ .
- A-deterministic: if  $pX \xrightarrow{a} q\beta$ ,  $pX \xrightarrow{a} r\gamma \in \mathbb{T}$ ,  $a \neq \epsilon$ , then  $q = r$  and  $\beta = \gamma$ .
- Deterministic: if the PDA is both  $\epsilon$ -deterministic and A-deterministic.
- Normed (or without redundancy): for any configuration  $q\beta$  of a graph  $G(p\alpha)$  there is a word  $u \in A^*$  and a state  $q$  such that  $p\alpha \xrightarrow{u} q\epsilon$ . One way of guaranteeing normedness is to ensure that for any state  $p$  and stack symbol  $X$  there is a word  $u \in A^*$  and a state  $q$  such that  $pX \xrightarrow{u} q\epsilon$ .

There is an important normal form in the case when a PDA is  $\epsilon$ -deterministic that  $\epsilon$ -transitions only pop the stack: if  $pX \xrightarrow{\epsilon} q\beta$  then  $\beta = \epsilon$ . By equivalent we mean that their collapsed graphs are isomorphic. There are three cases to consider.

1. If  $pX \xrightarrow{\epsilon} p_1X_1\alpha_1 \xrightarrow{\epsilon} p_2X_2\alpha_2 \xrightarrow{\epsilon} \dots$  then in the collapsed graph no configuration of the form  $pX\beta$  occurs because it is unstable. Therefore we delete from  $\mathbb{T}$  the transition  $pX \xrightarrow{\epsilon} p_1X_1\alpha_1$  and any transition of the form  $qY \xrightarrow{a} pX\alpha$  for  $a \in A \cup \{\epsilon\}$ .
2. If  $pX \xrightarrow{\epsilon} p_1X_1\alpha_1 \xrightarrow{\epsilon} \dots \xrightarrow{\epsilon} p_nX_n\alpha_n$  and  $p_nX_n\alpha_n$  is stable then we remove  $pX \xrightarrow{\epsilon} p_1X_1\alpha_1$  from  $\mathbb{T}$  and for each transition  $p_nX_n \xrightarrow{a} q\beta \in \mathbb{T}$  add the transition  $pX \xrightarrow{a} q\beta\alpha_n$  to  $\mathbb{T}$ .
3. If  $pX \xrightarrow{\epsilon} p_1X_1\alpha_1 \xrightarrow{\epsilon} \dots \xrightarrow{\epsilon} p_n\epsilon$  then we remove the transition  $pX \xrightarrow{\epsilon} p_1X_1\alpha_1$  from  $\mathbb{T}$  and add the new transition  $pX \xrightarrow{\epsilon} p_n\epsilon$  to  $\mathbb{T}$ .

In particular, this is a normal form for a DPDA (a *deterministic* pushdown automaton).

## 1.4 Decision questions

There are various decision questions that one can ask about PDA. One kind is essentially *graph-theoretic*: for instance, given configurations  $p\alpha$  and  $q\beta$ , is there a path from  $p\alpha$  to  $q\beta$  in the graph  $\mathbf{G}(p\alpha)$ ? In fact, as Büchi noted in the 1960s, there is an effective method for answering this question because the set of reachable configurations from any regular set of configurations is again a regular set (that can be characterized effectively using a finite-state automaton).

Another kind of decision question is *model checking*. The rooted graphs  $\mathbf{G}(p\alpha)$  (for a real-time PDA) and  $\mathbf{G}^c(q\beta)$  can be viewed as models with respect to which logical formulae are defined. A classical instance is *monadic second-order* logic which is first-order logic over these graphs (containing equality and an atomic binary predicate  $E_a$  for each  $a \in A$  with interpretation  $\xrightarrow{a}$ ) together with quantification over sets of vertices. Given a (collapsed) PDA graph and a formula  $\Phi(x)$  with one free variable, the decision question is whether  $\Phi(x)$  is true at the root of the graph. There is a significant history associated with this problem. In effect, Büchi showed that this question is decidable for example 1 of section 1.1 [3] and Rabin showed that it is decidable for example 2 [23]. The graphs  $\mathbf{G}(p\alpha)$  for real-time PDA exhibit a “regular” structure, as originally defined by Muller and Schupp [21]. From this they showed using Rabin’s result that the question is decidable for any real-time PDA graph. Finally, Caucal [9] extended the result to any collapsed graph  $\mathbf{G}(p\alpha)$  of a PDA. Standard temporal logics, LTL, CTL, CTL\* and modal  $\mu$ -calculus, are sublogics of monadic second-order logic, so model checking them on these infinite-state transition graphs is decidable. In practice, model checking these temporal logics appeal directly to automata (instead of monadic second-order logic).

Caucal also provided other characterisations of the graphs of pushdown automata. First he showed that the graphs of real-time PDA can be defined in terms of “pattern graphs” using deterministic graph grammars [7]. Secondly they are isomorphic to the transition graphs generated by Type0 grammars under prefix rewriting. Assume a finite family of nonterminals  $\mathbf{N}$  and a finite alphabet  $\mathbf{A}$ . A Type0 grammar under prefix rewriting is given by a finite family of basic transitions of the form  $\alpha \xrightarrow{a} \beta$  where  $\alpha$  and  $\beta$  belong to  $\mathbf{N}^*$ . The transition graph generated by a configuration  $\delta \in \mathbf{N}^*$  is determined by the basic transitions together with the prefix rule if  $\alpha \xrightarrow{a} \beta$  then  $\alpha\gamma \xrightarrow{a} \beta\gamma$  for any  $\gamma \in \mathbf{N}^*$ . He also defined a more general class of graphs, the *prefix-recognisable* graphs [9], where the finite set

of basic transitions are given by more general rules  $E \xrightarrow{a} F$  where  $E$  and  $F$  are regular expressions over the alphabet of nonterminals. It turns out that these graphs coincide with the collapsed graphs of PDA. For instance, the graph of figure 4 is (isomorphic to)  $G(X)$  when  $X \xrightarrow{a} Y^*$  and  $Y \xrightarrow{b} \epsilon$ .

A third kind of decision question deals with *equivalence checking*. Classically, formal language theory views the language generable from a configuration as paramount. Therefore, the language equivalence problem is: given two configurations of a PDA<sup>1</sup> do they recognise the same language?

**Definition 1** The *language* of  $p\alpha$ ,  $L(p\alpha) = \{w \in A^* : p\alpha \xrightarrow{w} q\epsilon \text{ for some } q \in P\}$ .

When recognising any such word the stack is thereby emptied. For instance,  $L(pX)$  in the case of example 3 of section 1.1 is the set of words  $\{a^n b^{n+1} : n \geq 0\} \cup \{a^n b : n \geq 0\}$ . This is called *empty stack acceptance*. A word  $w \in A^*$  is in  $L(p\alpha)$  if there is a  $w$ -path from  $p\alpha$  to a terminal state  $q\epsilon$  for some  $q$  in the *collapsed* graph  $G(p\alpha)$ .

The class of languages recognised by PDA is the context-free languages. It was shown in the 1960s that the language equivalence problem is undecidable (even for real-time single-state PDA). It was also shown that the language containment problem is undecidable for real-time single state DPDA<sup>2</sup>. (Visibly pushdown automata are real-time PDA that obey the following constraint: for every  $a \in A$ , if  $pX \xrightarrow{a} q\alpha \in T$  and  $p'Z \xrightarrow{a} q'\beta$  then  $|\alpha| = |\beta|$ . The language containment problem is decidable for such PDA [1]).

## 1.5 Bisimulation equivalence

Language based equivalences are not the only notion of equivalence between configurations of pushdown automata. There are proposals for finer equivalences in concurrency theory, where the behaviour of a process is presented as a labelled transition graph. There is a need for more intensional equivalences because language equivalence is not preserved by communicating automata. A pivotal notion, due to Park and Milner, is bisimulation equivalence.

Bisimulation equivalence is based on the existence of a relation between vertices of a transition graph which is preserved by transitions. More precisely a bisimulation relation is defined as follows.

**Definition 1** A binary relation  $R$  between vertices of transition graphs is a *bisimulation* relation provided that whenever  $(E, F) \in R$ , for all  $a \in A$ ,

$$\begin{aligned} \text{if } E \xrightarrow{a} E' \text{ then there is an } F'. F \xrightarrow{a} F' \text{ and } (E', F') \in R, \\ \text{if } F \xrightarrow{a} F' \text{ then there is an } E'. E \xrightarrow{a} E' \text{ and } (E', F') \in R. \end{aligned}$$

**Definition 2**  $E$  is *bisimulation equivalent* to  $F$ ,  $E \sim F$ , if there is a bisimulation relation containing  $(E, F)$ .

**Example 1** Consider the following three graphs.

$$\begin{array}{ccccc} p_1X & \xrightarrow{b} & p_1\epsilon & & q'_1\epsilon & \xleftarrow{b} & q_1X & \xrightarrow{b} & q_1\epsilon & & r_1\epsilon \\ \uparrow a & & & & & & \uparrow a & & & & \uparrow b \\ pX & & & & & & qX & & rX & \xrightarrow{a} & r_1X \\ \downarrow a & & & & & & \downarrow a & & & & \downarrow c \\ p_2X & \xrightarrow{c} & p_2\epsilon & & q'_2\epsilon & \xleftarrow{c} & q_2X & \xrightarrow{c} & q_2\epsilon & & r_2\epsilon \end{array}$$

Here  $pX \sim qX$ . The bisimulation relation  $R$  which witnesses this is

<sup>1</sup>As the disjoint union of two PDAs is a single PDA, we can restrict equivalence problems to configurations of a single PDA.

<sup>2</sup>That is, simple grammars that are defined later.

$$\{(pX, qX), (p_1X, q_1X), (p_2X, q_2X), (p_1\epsilon, q'_1\epsilon), (p_1\epsilon, q_1\epsilon), (p_2\epsilon, q'_2\epsilon), (p_2\epsilon, q_2\epsilon)\}$$

The proof that  $R$  is a bisimulation relation is straightforward. For example, consider the pair  $(p_1X, q_1X)$ . There is a single transition  $p_1X \xrightarrow{b} p_1\epsilon$  which is matched by  $q_1X \xrightarrow{b} q'_1\epsilon$  because  $(p_1\epsilon, q'_1\epsilon) \in R$ . And there are two transitions  $q_1X \xrightarrow{b} q'_1\epsilon$  and  $q_1X \xrightarrow{b} q_1\epsilon$  which are both matched by the single transition  $p_1X \xrightarrow{b} p_1\epsilon$ , as both  $(p_1\epsilon, q'_1\epsilon), (p_1\epsilon, q_1\epsilon) \in R$ .

The vertices  $pX$  and  $rX$  are not bisimilar, even though they recognise the same language. Assume that there is a bisimulation relation  $R$  containing the pair  $(pX, rX)$ . The problem is how to match the transition  $rX \xrightarrow{a} r_1X$ . It can not be matched by  $pX \xrightarrow{a} p_1X$  because  $r_1X \xrightarrow{c} r_2\epsilon$  and it can not be matched by  $pX \xrightarrow{a} p_2X$  because  $r_1X \xrightarrow{b} r_1\epsilon$ .  $\square$

**Example 2** In the following  $pX \sim qY$ .

$$\begin{array}{ccccccc} pX & \xrightarrow{a} & pXX & \xrightarrow{a} & pXXX & \xrightarrow{a} & pXXXX & \xrightarrow{a} & \dots \\ qY & \xrightarrow{a} & q_1Y & \xrightarrow{a} & q_1YY & \xrightarrow{a} & q_1YYY & \xrightarrow{a} & \dots \end{array}$$

The bisimulation relation  $R$  which witnesses this equivalence is infinite.

$$\{(pX^{2n}, q_1Y^n) : n > 0\} \cup \{(pX^{2n+1}, qY^n) : n > 0\}$$

For instance  $(pX^{24}, q_1Y^{12}) \in R$ . There is a single transition  $pX^{24} \xrightarrow{a} pX^{25}$  and  $q_1Y^{12} \xrightarrow{a} qY^{13}$ , and  $(pX^{25}, qY^{13}) \in R$ .  $\square$

There is also a notion of approximation associated with bisimulation equivalence.

**Definition 2** The family  $\{\sim_n : n \geq 0\}$  between vertices of transition graphs is defined inductively as follows.

$$\begin{array}{l} E \sim_0 F \text{ for all vertices } E, F \\ E \sim_{n+1} F \text{ iff for all } a \in A \\ \quad \text{if } E \xrightarrow{a} E' \text{ then there is an } F'. F \xrightarrow{a} F' \text{ and } E' \sim_n F', \text{ and} \\ \quad \text{if } F \xrightarrow{a} F' \text{ then there is an } E'. E \xrightarrow{a} E' \text{ and } E' \sim_n F' \end{array}$$

For instance in the case of example 1,  $pX \sim_1 rX$  but  $pX \not\sim_2 rX$ .

**Fact 1** If  $p\alpha \sim q\beta$  then for all  $n \geq 0$ .  $p\alpha \sim_n q\beta$ .

The converse of fact 1 is not in general true for collapsed graphs  $G^c(p\alpha)$ . It is possible that  $E \not\sim F$  and  $E \sim_n F$  for all  $n \geq 0$ . (Find an example.) However, the converse does hold for graphs which have finite out degree.

**Fact 2** If the transition graphs containing  $E$  and  $F$  have finite out degree and for all  $n \geq 0$ .  $E \sim_n F$  then  $E \sim F$ .

Baeten, Bergstra and Klop showed in 1987 that bisimulation equivalence is decidable for normed, single-state real-time PDA [2]. This was extended to all single-state real-time PDA [11] and then to all real-time PDA and to the collapsed graphs of  $\epsilon$ -deterministic PDA [25]. However, the problem is undecidable for arbitrary collapsed graphs of PDA (a result that is a consequence of [28]).

## 1.6 The DPDA equivalence problem

A more classical definition of a PDA includes an extra component:  $F \subseteq P$  which are *final states*. The language of a configuration is then the set of words  $w \in A^*$  such that  $p\alpha \xrightarrow{w} q\beta$  where  $q \in F$ . This is called *final state acceptance*.

For pushdown automata, in general, the languages recognised under final state acceptance coincide with those recognised under empty stack acceptance. However, this is not true in the case of DPDA. The languages recognised by DPDA under final state acceptance are the *deterministic context-free* languages. The languages accepted under final state acceptance is a proper subset of these deterministic languages. However, we still work with empty stack acceptance because of the following. For any language  $L$  accepted by a DPDA configuration under final state acceptance, there is a DPDA configuration that accepts  $L\$$  where  $\$ \notin A$  is an end of word marker: the language  $L\$ = \{w\$ : w \in L\}$ . To see this, first include a new bottom of the stack symbol  $B$  in a DPDA with final state acceptance: so,  $p\alpha \xrightarrow{u} q\beta$  where  $q$  is a final state if, and only if,  $p\alpha B \xrightarrow{u} q\beta B$  in the amended DPDA. If for a final state  $q$  and stack symbol  $X$ ,  $qX \xrightarrow{\epsilon} r\epsilon$  and  $r$  is not final then introduce a new final state  $r'$  and replace this transition in  $\mathbb{T}$  with  $qX \xrightarrow{\epsilon} r'\epsilon$  and add transitions for  $r'$  that are the same as for  $r$ . Next, for each final state  $q$  and stack element  $S$ , if  $qS$  is stable then introduce a new basic transition  $qS \xrightarrow{\$} e\epsilon$  where  $e$  is a new state whose role is to erase all stack elements, so  $eS \xrightarrow{\epsilon} e\epsilon$  for all  $S$ . Clearly,  $p\alpha B \xrightarrow{u} q\beta B$  in the amended DPDA with final state acceptance if, and only if,  $p\alpha B \xrightarrow{u\$} e\epsilon$  in the DPDA that accepts with empty stack acceptance.

The DPDA equivalence problem was posed in the 1960s [12]: given two configurations  $p\alpha, q\beta$  of a DPDA, is there an effective procedure for deciding whether  $L(p\alpha) = L(q\beta)$ ? It was a celebrated open problem until it was solved positively by Geraud Sénizergues [24, 26]. It seems that the notation of pushdown configurations, although simple, is not rich enough to sustain a proof. Deeper algebraic structure needs to be exposed. Sénizergues's proof is primarily algebraic. This proof was simplified in [29, 27]. However, these solutions to the problem involve two semi-decision procedures: one to show inequivalence (“find a smallest distinguishing word”) the other to show equivalence (basically, using a *nondeterministic* proof procedure). Therefore, there is no complexity bound on the decision procedure.

A much simpler *deterministic* decision procedure with a primitive recursive upper bound on its complexity was presented in [30]. It is this proof that is the main topic of these notes. The proof technique is somewhat different from Sénizergues's method and is based on the *bisimulation equivalence problem* for deterministic graphs  $G^c(p\alpha)$ . However, essential elements of Sénizergues's proof are still present in the decision procedure.

## 2 Decidability of Equivalence of Simple Grammars

Surprisingly, the key to understanding the DPDA equivalence problem is real-time single-state PDA.

### 2.1 Pushdown grammars

A *pushdown grammar* (a PDG) is a real-time single state PDA with ingredients:

- A finite set of stack symbols  $S$
- A finite alphabet  $A$
- A finite set of basic transitions  $T$

A basic transition has the form  $X \xrightarrow{a} \alpha$  where  $X \in S$ ,  $a \in A$  and  $\alpha \in S^*$ . A configuration  $\beta$  is a sequence of stack symbols. The prefix rule is

$$\text{PRE if } X \xrightarrow{a} \alpha \in T \text{ then } X\delta \xrightarrow{a} \alpha\delta.$$

We assume that a PDG is in normal form. First, any transition  $X \xrightarrow{a} \alpha \in T$  has the property that  $|\alpha| \leq 2$ . To ensure this, we add extra stack symbols as outlined in section 1.1. Second, a PDG is *normed*: for every  $X \in S$  there is a word  $u$  such that  $u \in L(X)$ . We now show how to ensure this.

Assume a fixed total ordering  $<$  on  $A$ .

**Definition 1** The word  $u$  is “smaller” than  $v$ , if  $|u| < |v|$  or  $|u| = |v|$  and  $u$  is lexicographically less than  $v$  with respect to the ordering on  $A$ : that is, if  $|u| = |v|$ ,  $u = waw'$  and  $v = wbv'$  then  $a < b$ .

**Definition 2** For each stack symbol  $X$ , if  $L(X) \neq \emptyset$  then  $w(X)$  is the *smallest* word in  $\{u : X \xrightarrow{u} \epsilon\}$ . The *norm* of  $X$  is the length of  $w(X)$ ,  $|w(X)|$ .

It is easy to compute  $w(X)$ , if it exists, for each stack symbol  $X$ . Initially, stack symbols with norm 1 are identified by examining transitions in  $T$  of the form  $X \xrightarrow{a} \epsilon$ , and their shortest words are defined. There must be at least one stack symbol with norm 1 unless  $L(X) = \emptyset$  for all  $X \in S$ . Next, stack symbols with norm 2 are identified.  $X$  has norm 2 if  $X$  does not have norm 1 and  $X \xrightarrow{a} Z \in T$  and  $Z$  has norm 1. A stack symbol  $X$  has norm 3 if it does not have norm 1 or norm 2, and either  $X \xrightarrow{a} Z \in T$  and  $Z$  has norm 2 or  $X \xrightarrow{a} YZ \in T$  and both  $Y$  and  $Z$  have norm 1. Identification of norm is iterated, until either  $w(X)$  is calculated for each stack symbol  $X$  or  $X$  can not have a norm: the current largest norm is  $k$  and no stack symbol has norm between  $k$  and  $2k + 1$ .

If this procedure leaves  $X$  unnormed then it is deleted from  $S$ , and all transitions  $Z \xrightarrow{a} \alpha \in T$  with  $X = Z$  or  $\alpha$  containing  $X$  are deleted from  $T$ . The result is a normed PDG that preserves language equivalence (for configurations  $\alpha$  such that  $L(\alpha) \neq \emptyset$ ).

An important measure of a PDG (in normal form) is its maximum norm  $M$ .

**Definition 3**  $M = \max\{|w(X)| : X \in S\}$ .

In the worst case,  $M$  is exponential in the number of stack symbols. If  $S = \{X_1, \dots, X_n\}$  and  $T$  is,

$$X_n \xrightarrow{a} \epsilon \quad X_{n-1} \xrightarrow{a} X_n X_n \quad X_{n-2} \xrightarrow{a} X_{n-1} X_{n-1} \quad \dots \quad X_1 \xrightarrow{a} X_2 X_2,$$

then  $w(X_{n-j}) = a^{2^j}$ .

The definition of norm extends to configurations of a PDG.

**Definition 4** For each configuration  $\alpha$ , the word  $w(\alpha)$  is the unique smallest word  $v$  such that  $\alpha \xrightarrow{v} \epsilon$ . The norm of  $\alpha$  is  $|w(\alpha)|$ .

**Fact 1**

1.  $w(\epsilon) = \epsilon$  and  $|w(\epsilon)| = 0$
2.  $w(X\alpha) = w(X)w(\alpha)$  and  $|w(X\alpha)| = |w(X)| + |w(\alpha)|$

As a first step towards solving the DPDA equivalence problem, we prove decidability of language equivalence for simple grammars, which was first shown by Korenjak and Hopcroft in 1966 using a method which is not so dissimilar from that presented here [20].

## 2.2 Simple grammars

A simple grammar is a deterministic PDG. The key restriction is

- Determinism: if  $X \xrightarrow{a} \alpha \in \mathbb{T}$  and  $X \xrightarrow{a} \beta \in \mathbb{T}$  then  $\alpha = \beta$

The recognition power of simple grammars is less than that of DPDA. For example, the language  $L = \{a^n b^n : n > 0\} \cup \{a^n c : n > 0\}$  is generable by a DPDA configuration: it is  $L(pX)$  of example 3 of section 1.1 when the transition  $pX \xrightarrow{b} r\epsilon$  is changed to  $pX \xrightarrow{c} r\epsilon$ . However, it is not generable by a configuration of a simple grammar (why?).

**Example 1** Let  $\mathbb{T}$  be

$$\begin{array}{lll} X \xrightarrow{a} YX & X \xrightarrow{b} \epsilon & Y \xrightarrow{b} X \\ A \xrightarrow{a} C & A \xrightarrow{b} \epsilon & C \xrightarrow{b} AA. \end{array}$$

The transition graphs  $G(X)$  and  $G(A)$  are pictured in figure 5. Here,  $w(X) = b$  and  $w(A) = b$ . Because of the transition  $Y \xrightarrow{b} X$ , the symbol  $Y$  has norm 2 and  $w(Y) = bb$ . The symbol  $C$  has norm 3 because  $C \xrightarrow{b} AA$ , so  $w(C) = bbb$ . So the maximum norm  $M = 3$ .  $\square$

Configurations of a simple grammar are sequences of stack symbols. An extra configuration, the deadlocked configuration  $\emptyset$ , is added for the purpose of a useful notation<sup>3</sup>, “the result of  $\alpha$  after the word  $u$ ,” written  $\alpha \cdot u$ . The configuration  $\alpha \cdot u$  is either the configuration  $\beta$  such that  $\alpha \xrightarrow{u} \beta$  or it is the deadlocked configuration  $\emptyset$ . The configuration  $\emptyset$  is unnormed, its size,  $|\emptyset|$ , is 0 and  $L(\emptyset) = \emptyset$ . It is assumed that  $\emptyset\alpha = \emptyset = \alpha\emptyset$ .

**Fact 1** If  $\alpha \in S^* \cup \{\emptyset\}$ , then

1.  $(\alpha \cdot \epsilon) = \alpha$
2.  $(\emptyset \cdot au) = \emptyset$
3. if  $X \xrightarrow{a} \beta \in \mathbb{T}$ , then  $(X\alpha \cdot au) = (\beta\alpha \cdot u)$
4. if there is no transition  $X \xrightarrow{a} \beta \in \mathbb{T}$ , then  $(X\alpha \cdot au) = \emptyset$ .

<sup>3</sup> $\emptyset$  has an important role later when configurations are extended to sets of sequences of stack symbols  $\{\alpha_1, \dots, \alpha_n\}$ , and it is then genuinely the emptyset.

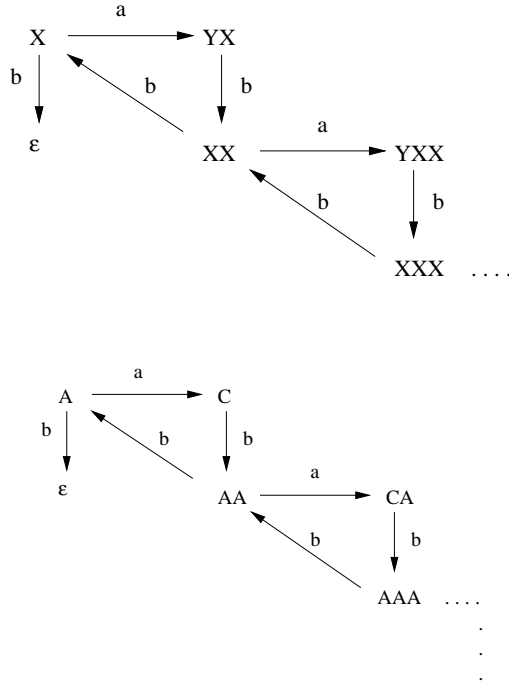


Figure 5:  $G(X)$  and  $G(A)$

The inclusion of  $\emptyset$  guarantees that  $(\alpha \cdot u)$  is always defined, and is unique because simple grammars are deterministic. In example 1, above,  $(YX \cdot bab) = XXX$  and  $(AA \cdot aa) = \emptyset$ . Some obvious properties of “after” are now listed.

**Fact 2**

1. If  $(\alpha \cdot u) = \emptyset$  then  $(\alpha \cdot uv) = \emptyset$
2. If  $(\alpha \cdot u) = \epsilon$  then  $(\alpha \cdot ua) = \emptyset$
3.  $\max\{0, |\alpha| - |u|\} \leq |(\alpha \cdot u)| \leq |\alpha| + |u|$
4.  $(\alpha \cdot uv) = (\alpha \cdot u) \cdot v$
5.  $(\alpha\beta \cdot w(\alpha)) = \beta$
6. If  $|u| \leq |w(\alpha)|$  then  $(\alpha\beta \cdot u) = (\alpha \cdot u)\beta$

The language of a configuration  $L(\alpha)$  is *prefix-free* because of determinism: if a word belongs to  $L(\alpha)$ , then no proper prefix belongs to it.

**Fact 3** If  $u \in L(\alpha)$ , then for any  $v \in A^+$ ,  $uv \notin L(\alpha)$ .

The decision problem for simple grammars is: given configurations  $\alpha, \beta \in S^*$ , is  $L(\alpha) = L(\beta)$ ? (The decision problem  $L(\alpha) \subseteq L(\beta)$ ? is *undecidable*.) Because simple grammars are deterministic (and normed) language equivalence coincides with bismulation equivalence.

**Fact 4**  $L(\alpha) = L(\beta)$  iff  $\alpha \sim \beta$ .

**Example 2**  $L(A) = L(X)$  where  $X$  and  $A$  are from example 1, so  $A \sim X$ . However, the bismulation  $R$  justifying equivalence is infinite.

$$R = \{(X^n, A^n) : n \geq 0\} \cup \{(YX^{n+1}, CA^n) : n \geq 0\}$$

UNF

$$\frac{\alpha \doteq \beta}{(\alpha \cdot a_1) \doteq (\beta \cdot a_1) \dots (\alpha \cdot a_k) \doteq (\beta \cdot a_k)} \mathbf{A} = \{a_1, \dots, a_k\}$$

Figure 6: The rule UNF

$R$  obeys the hereditary conditions for being a bisimulation. For instance, the pair  $(YX^8, CA^7) \in R$  and  $YX^8 \xrightarrow{b} X^9$  and  $CA^7 \xrightarrow{b} A^9$ , and  $(X^9, A^9) \in R$ .  $\square$

Example 2 illustrates that a justifying bisimulation relation may be infinite, so it is necessary to supply a more sophisticated technique than merely exhibiting a bisimulation witness if we wish to solve the decision problem for simple grammars.

### 2.3 Tableau proof rules

The decision procedure for simple grammars is a *tableau proof system*, consisting of proof rules which allow goals to be reduced to subgoals. Goals and subgoals are all of the form  $\alpha \doteq \beta$ , “is  $\alpha \sim \beta$ ?”, where  $\alpha$  and  $\beta$  are configurations of a simple grammar.

The initial tableau proof rule is UNF (unfold) in figure 6. The goal,  $\alpha \doteq \beta$  reduces to the subgoal  $(\alpha \cdot a) \doteq (\beta \cdot a)$  for each  $a \in \mathbf{A}$ . The application of this simple rule is both “complete” and “sound”. Completeness is the property that if the goal,  $\alpha \doteq \beta$ , is true then so are all the subgoals,  $(\alpha \cdot a_i) \doteq (\beta \cdot a_i)$ .

**Fact 1** *If  $\alpha \sim \beta$ , then for all  $a \in \mathbf{A}$ ,  $(\alpha \cdot a) \sim (\beta \cdot a)$ .*

Soundness is the converse, that if all the subgoals are true then so is the goal which is equivalent to, if the goal is false,  $\alpha \not\sim \beta$ , then so is at least one of the subgoals. However, there is a finer account that uses approximants.

**Fact 2** *If  $\alpha \sim_n \beta$  and  $\alpha \not\sim_{n+1} \beta$ , then for some  $a \in \mathbf{A}$ ,  $(\alpha \cdot a) \not\sim_n (\beta \cdot a)$ .*

**Example 1** In the case of the simple grammar of example 1 of section 2.2, the following is an application of UNF to the goal  $A \doteq X$ .

$$\frac{A \doteq X}{C \doteq YX \quad \epsilon \doteq \epsilon}$$

$C \doteq YX$ , is  $(A \cdot a) \doteq (X \cdot a)$  and  $\epsilon \doteq \epsilon$  is  $(A \cdot b) \doteq (X \cdot b)$ .  $\square$

Repeated applications of UNF continually reduce goals. For instance, the following tree of applications of UNF starts with goal  $A \doteq X$  of the example, above.

$$\frac{\frac{\frac{A \doteq X}{C \doteq YX} \text{ UNF} \quad \epsilon \doteq \epsilon}{\emptyset \doteq \emptyset} \text{ UNF} \quad AA \doteq XX}{CA \doteq YXX \quad A \doteq X} \text{ UNF}$$

In each application of UNF, the left subgoal is the result after  $a$  and the right subgoal is the result after  $b$ . If  $\alpha' \doteq \beta'$  is a subgoal which is the result of  $m$  consecutive applications of UNF from the goal  $\alpha \doteq \beta$ , then there is a word  $u$  such that  $|u| = m$  and  $\alpha' = (\alpha \cdot u)$  and  $\beta' = (\beta \cdot u)$ . The word  $u$  is then said to be the associated word with this sequence of applications. In the example, above,  $CA \doteq YXX$  is the

BAL(L) and BAL(R)

$$\begin{array}{ccc}
X\alpha \doteq \beta & & \beta \doteq X\alpha \\
\vdots & \text{C} & \vdots \\
\alpha'\alpha \doteq \beta' & & \beta' \doteq \alpha'\alpha \\
\hline
\alpha'(\beta \cdot w(X)) \doteq \beta' & & \beta' \doteq \alpha'(\beta \cdot w(X))
\end{array}$$

where C is the condition

1.  $|\alpha| > 0$  and  $|\alpha'| > 0$ , and
2. there are precisely  $|w(X)|$  consecutive applications of UNF between the top goal,  $X\alpha \doteq \beta$  ( $\beta \doteq X\alpha$ ), and the bottom goal,  $\alpha'\alpha \doteq \beta'$  ( $\beta' \doteq \alpha'\alpha$ ), and no applications of any other rule.

Figure 7: The balance rules

result of 3 consecutive applications of UNF from  $A \doteq X$ . The associated word is  $aba$ .

A key idea is reduction of a goal to a “balanced” subgoal, and this is the role of the rules BAL(L), balance left, and BAL(R), balance right, in figure 7. Balance rules were introduced by Sénizergues [26] for the more general DPDA equivalence problem<sup>4</sup>: in his framework, they are the transformations  $T_B$ . The imbalance of a goal  $\alpha \doteq \beta$  is the length of the largest prefix of  $\alpha$  or  $\beta$  before they have a common tail. If  $\alpha = \alpha_1\delta$  and  $\beta = \beta_1\delta$  and the only common suffix of  $\alpha_1$  and  $\beta_1$  is the empty sequence, then the imbalance between  $\alpha$  and  $\beta$  is  $\max\{|\alpha_1|, |\beta_1|\}$ . If the imbalance between  $\alpha$  and  $\beta$  is 0, then  $\alpha = \beta$ , so  $\alpha \sim \beta$ .

Assume a goal  $\beta \doteq X\alpha$ , the initial goal of a BAL(R), and assume that  $\beta = Y_1 \dots Y_n$  and  $n \geq M+1$ . Also, assume that there are  $|w(X)|$  consecutive applications of UNF between this goal and the goal  $\beta' \doteq \alpha'\alpha$ , and  $|\alpha'| > 0$ . Therefore, there is a word  $u$  of size  $|w(X)|$  associated with this sequence of applications of UNF, and  $(\beta \cdot u) = \beta'$  and  $(X\alpha \cdot u) = \alpha'\alpha = (X \cdot u)\alpha$  because  $|u| = |w(X)|$ . Clearly,  $|u| \leq M$ , so  $(\beta \cdot u) = (Y_1 \dots Y_M \cdot u)Y_{M+1} \dots Y_n$ . An application of BAL(R) to  $\beta' \doteq \alpha'\alpha$  has result  $\beta' \doteq \alpha'(\beta \cdot w(X))$  which is therefore the goal

$$(Y_1 \dots Y_M \cdot u)Y_{M+1} \dots Y_n \doteq \alpha'(Y_1 \dots Y_M \cdot w(X))Y_{M+1} \dots Y_n.$$

However,

$$\begin{array}{rcl}
|(Y_1 \dots Y_M \cdot u)| & \leq & 2M \\
|\alpha'| = |(X \cdot u)| & \leq & M+1 \\
|(Y_1 \dots Y_M \cdot w(X))| & \leq & 2M
\end{array}$$

and therefore, the maximum imbalance of the subgoal is  $3M+1$ . This bound is independent of the sizes of, and the imbalance between, the configurations in the goals of the rule.

Completeness of an application of BAL is that if the goals are true then so is the subgoal<sup>5</sup>: its proof depends on the following two properties.

<sup>4</sup>We shall use the more general rules later.

<sup>5</sup>A more parsimonious claim is that if the initial goal of an application of BAL is true then so is the main goal and the subgoal.

CUT

$$\frac{\alpha\delta \doteq \beta\delta}{\alpha \doteq \beta} \delta \neq \epsilon$$

Figure 8: The rule CUT

**Proposition 1**

1. If  $X\alpha \sim \beta$  then  $\alpha \sim (\beta \cdot w(X))$ .
2. If  $\alpha \sim \beta$  then  $\delta\alpha \sim \delta\beta$ .

**Proof:** If  $X\alpha \sim \beta$  then  $X\alpha \xrightarrow{w(X)} \alpha$  and therefore  $\beta \xrightarrow{w(X)} \beta'$  and  $\alpha \sim \beta'$ . By definition  $\beta' = \beta \cdot w(X)$ . For 2 if  $R$  is a bisimulation containing the pair  $(\alpha, \beta)$  then  $R \cup \{(\gamma\alpha, \gamma\beta) : \gamma \in \mathbf{S}^*\}$  is also a bisimulation.  $\square$

**Fact 3** If  $X\alpha \sim \beta$  and  $\alpha'\alpha \sim \beta'$  then  $\alpha'(\beta \cdot w(X)) \sim \beta'$ .

Soundness utilises the following properties.

**Proposition 2**

1. If  $X\alpha \sim_n \beta$  and  $n \geq |w(X)|$  then  $\alpha \sim_{n-|w(X)|} (\beta \cdot w(X))$ .
2. If  $\alpha \sim_n \beta$  and  $|\delta| \geq k$  then  $\delta\alpha \sim_{n+k} \delta\beta$ .

**Proof:** If  $X\alpha \sim_n \beta$  and  $n > |w(X)|$  then  $X\alpha \xrightarrow{w(X)} \alpha$  and therefore  $\beta \xrightarrow{w(X)} \beta'$  and  $\alpha \sim_{n-|w(X)|} \beta'$ . By definition  $\beta' = \beta \cdot w(X)$ . For 2 consider any word  $u$  such that  $|u| \leq n+k$ . Assume  $\delta\alpha \cdot u$  is defined. We therefore need to show that  $\delta\beta \cdot u$  is also defined. There are two cases. First  $\delta\alpha \cdot u = (\delta \cdot u)\alpha$ , and so  $\delta\beta \cdot u = (\delta \cdot u)\beta$ . Second  $u = v\delta$  and  $\delta \cdot v = \epsilon$ , but then  $|v| \geq k$  and so  $|w| \leq n$ . Consequently  $\delta\beta \cdot u$  is  $\beta \cdot w$  which is defined because  $\alpha \sim_n \beta$ .  $\square$

In fact soundness is subtle. We need to bear in mind “global” soundness of the tableau construction. The main idea, as we shall see, is that if there is a successful tableau whose root is false then there is an offending path of false goals. This idea is refined using approximants. If a goal  $\gamma \doteq \delta$  is false then there is a least  $n$  such that  $\gamma \not\sim_n \delta$ . The falsity index decreases through an application of UNF, as shown by fact 2. Consequently, if a successful tableau has a false root then there is an offending path of goals whose “falsity” index is decreasing whenever UNF is applied.

Consider the circumstance when the initial goal of an application of BAL is false,  $X\alpha \not\sim \beta$ . There is a least  $n+1$  such that  $X\alpha \not\sim_{n+1} \beta$  and  $X\alpha \sim_n \beta$ . Assume that  $m = |w(X)|$ . For soundness of an application of BAL we are only interested in the case when there is an offending path of false goals whose falsity index decreases as UNF is applied which passes through the main goal of the application, and so  $n \geq m$ . In which case it follows that  $\alpha_1\alpha \not\sim_{(n+1)-m} \beta'$  because there are exactly  $m$  applications of BAL between these goals. The soundness property we wish to show is that this falsity index is preserved by the application of the rule.

**Fact 4** If  $X\alpha \sim_n \beta$  and  $|w(X)| = m$  and  $n > m$  and  $\alpha'\alpha \not\sim_{(n+1)-m} \beta'$  and  $|\alpha'| > 0$  then  $\alpha'(\beta \cdot w(X)) \not\sim_{(n+1)-m} \beta'$ .

The final rule in figure 8, CUT, allows common tails to be “cut” from a goal.

**Proposition 3** If  $\alpha\delta \sim \beta\delta$  then  $\alpha \sim \beta$ .

### Successful final goals

$$\begin{array}{ccc} & \alpha \doteq \beta & \\ & \vdots & \text{UNF at least once} \\ \alpha \doteq \alpha & \alpha \doteq \beta & \end{array}$$

### Unsuccessful final goals

$$\alpha \doteq \beta \quad (\text{exactly one of } \alpha, \beta \text{ is } \emptyset)$$

Figure 9: Final goals

**Proof:** Assume  $\alpha\delta \sim \beta\delta$  and consider a smallest word  $w(\delta)$  recognised by  $\delta$ , so  $\delta \xrightarrow{w(\delta)} \epsilon$  and if  $\delta \xrightarrow{u} \epsilon$  then  $|u| \geq |w(\delta)|$ . Assume that  $\alpha \not\sim \beta$ . There is therefore a smallest word which distinguishes between them, say  $v$ . Without loss of generality assume  $v$  is recognised by  $\alpha$ ,  $\alpha \xrightarrow{v} \epsilon$ , but  $\beta \cdot v \neq \epsilon$ . However  $\alpha\delta \xrightarrow{vw(\delta)} \epsilon$ . Because  $\beta\delta \sim \alpha\delta$  it follows that  $\beta\delta \xrightarrow{vw(\delta)} \epsilon$ . One possibility is  $w(\delta) = w_1w_2$  and  $w_2 \neq \epsilon$  and  $\beta \xrightarrow{vw_1} \epsilon$  and  $\delta \xrightarrow{w_2} \epsilon$ , but this contradicts that  $w(\delta)$  is a smallest word recognised by  $\delta$ . Otherwise  $v = v_1v_2$  and  $v_2 \neq \epsilon$  and  $\beta \xrightarrow{v_1} \epsilon$  and  $\delta \xrightarrow{v_2w(\delta)} \epsilon$ . But this contradicts that  $v$  is a smallest word which distinguishes  $\alpha$  and  $\beta$  because  $\alpha \xrightarrow{v_1} \alpha'$  where  $\alpha' \neq \epsilon$ . Therefore  $\alpha \sim \beta$ .  $\square$

If the goal of an application of CUT is false then so is its subgoal (and it preserves the “falsity” index).

**Proposition 4** *If  $\alpha\delta \not\sim_n \beta\delta$  then  $\alpha \not\sim_n \beta$ .*

**Proof:** We show its equivalent, if  $\alpha \sim_n \beta$  then  $\alpha\delta \sim_n \beta\delta$  by induction on  $n$ . The base case  $n = 0$  is clear. Assume it holds for  $n < k$  and consider  $n = k$ . Assume  $\alpha \sim_n \beta$  and  $\alpha\delta \xrightarrow{a} \alpha'$ . If  $\alpha = \epsilon$  and  $\delta \xrightarrow{a} \alpha'$  then either  $n = 0$  and the result already follows or  $\beta = \epsilon$  and therefore  $\beta\delta \xrightarrow{a} \alpha'$  and clearly  $\alpha' \sim_{n-1} \alpha'$ . If  $\alpha \xrightarrow{a} \alpha_1$  and  $\alpha' = \alpha_1\delta$  then  $\beta \xrightarrow{a} \beta_1$  and  $\alpha_1 \sim_{n-1} \beta_1$  and therefore by the induction hypothesis  $\alpha_1\delta \sim_{n-1} \beta_1\delta$ . The symmetric case  $\beta\delta \xrightarrow{a} \beta'$  is similar.  $\square$

## 2.4 Successful tableaux

In the previous section we presented and justified four tableau proof rules. We now show that these rules lead to an effective decision procedure for checking equivalence of configurations of a simple grammar.

A missing ingredient in the tableau description is when a current goal is final. Final goals are either *successful* or *unsuccessful*, and are presented in figure 9. A successful final goal is either an identity or a repeat goal with at least one application of UNF between the repetition. An identity is clearly true: for the argument why a repeat is successful see theorem 2 of the later section 2.5. An unsuccessful final goal  $\alpha \doteq \beta$  is clearly false, as one and only one configuration is  $\emptyset$ , and therefore  $L(\alpha) \neq L(\beta)$ .

The tableau procedure starts with an initial goal,  $\alpha \doteq \beta$ , “is  $\alpha \sim \beta$ ?”, and one then builds a proof tree by applying the tableau rules. Goals are thereby reduced to subgoals. Rules are not applied to final goals.

**Definition 1** A *successful* tableau for  $\alpha \doteq \beta$  is a finite proof tree with root  $\alpha \doteq \beta$  and all of whose leaves are successful final goals. Otherwise a tableau is *unsuccessful*: that is, if it is not a finite proof tree or if it contains an unsuccessful final goal.

**Example 1** Earlier we proved that  $A \sim X$  in example 2 of section 2.1. The set  $\mathbb{T}$  of transitions are as follows.

$$\begin{array}{ccc} A \xrightarrow{a} C & A \xrightarrow{b} \epsilon & C \xrightarrow{b} AA \\ X \xrightarrow{a} YX & X \xrightarrow{b} \epsilon & Y \xrightarrow{b} X \end{array}$$

Below is a successful tableau for  $A \doteq X$ .

$$\frac{\frac{\frac{A \doteq X}{C \doteq YX} \text{ UNF}}{AA \doteq XX} \text{ UNF}}{\frac{CA \doteq YXX}{CX \doteq YXX} \text{ BAL(L)}} \text{ UNF} \quad \frac{A \doteq X}{\epsilon \doteq \epsilon} \text{ UNF}$$

$$\frac{\frac{CA \doteq YXX}{CX \doteq YXX} \text{ BAL(L)}}{C \doteq YX} \text{ CUT}$$

The annotation explains which rules are applied. The initial goal  $A \doteq X$  reduces to the subgoals  $(A \cdot a) \doteq (X \cdot a)$  and  $(A \cdot b) \doteq (X \cdot b)$ . The second of these goals is a final successful goal,  $\epsilon \doteq \epsilon$ . There are two further applications of UNF to the first of these goals. The right subgoal  $A \doteq X$  is a successful final goal because it is a repeat and there is at least one application of UNF between the repetition. BAL(L) is applied to the goal  $CA \doteq YXX$  with initial premise  $AA \doteq XX$ , as follows.

$$\frac{\frac{AA \doteq XX}{\vdots}{CA \doteq YXX}}{C(XX \cdot b) \doteq YXX}$$

This fulfills the conditions of the application of BAL(L),  $w(A) = b$  and so  $|w(Y)| = 1$  and there is precisely 1 application of UNF between the two premises and no other rule is applied. The left configuration has changed from  $AA$  to  $CA$  in the second goal, and therefore the rule sanctions replacement of the final occurrence of  $A$  with  $(XX \cdot w(A))$  which is  $X$ . CUT is then applicable to the goal  $CX \doteq YXX$  and the result is a final successful goal  $C \doteq YX$  because it is a repeat.  $\square$

In example 1 there is an application of BAL(L) to the goal  $CA \doteq YXX$ . However BAL(R) also applies to this goal in two distinct ways (which in fact produce the same subgoal).

$$\frac{\frac{AA \doteq XX}{\vdots}{CA \doteq YXX}}{CA \doteq YX(AA \cdot b)} \quad \frac{\frac{C \doteq YX}{\vdots}{CA \doteq YXX}}{CA \doteq YX(C \cdot bb)}$$

The difference in application is that  $w(Y) = bb$ , whereas  $w(X) = b$ .

**Example 2** Example 1 illustrates a successful tableau. However the rules allow tableaux that do not terminate, for instance, by only applying UNF.

$$\begin{array}{c}
\frac{A \doteq X}{\frac{C \doteq YX}{AA \doteq XX} \text{ UNF}} \text{ UNF} \quad \epsilon \doteq \epsilon \\
\frac{CA \doteq YXX}{AAA \doteq XXX} \text{ UNF} \quad A \doteq X \\
\vdots \quad \vdots
\end{array}$$

There is an infinite path of goals  $A^n \doteq X^n$  for all  $n > 0$ . □

A simple grammar is deterministic, and therefore we would prefer that there is just one tableau for any goal. To achieve uniqueness of tableau, we appeal to the ordering on  $\mathbf{A}$  (see definition 1 of section 2.1): that in an application of UNF, the subgoals are ordered relative to this ordering. In example 1 above we assume that  $a < b$ .

Next we provide a strategy for applying tableau proof rules. The important issue is when to apply the BAL rules and with respect to which premises. We assume that a BAL rule applies to a goal using the premise above which is the closest: in the case of the two applications of BAL(R) above we assume it is the first application only which is allowed. We then place the following priority on the tableau rules.

1. Apply BAL(L) followed immediately, if applicable, by CUT, or
2. Apply BAL(R) followed immediately, if applicable, by CUT, or
3. Apply UNF

Given a goal one tries first to apply BAL(L), and if it is not applicable then one tries to apply BAL(R). Only if one of the BAL rules does not apply does one apply UNF. CUT is only applied after a BAL. The potentially infinite tableau of example 2 is thereby excluded, because BAL(L) applies to the goal  $CA \doteq YXX$ . The tableau of example 1 follows this strategy, and it is therefore the unique tableau for the initial goal  $A \doteq X$ .

## 2.5 Correctness of tableaux

We now come to the main results, which show decidability of language equivalence for simple grammars.

**Theorem 1** *There is a unique finite tableau for goal  $\alpha \doteq \beta$ .*

**Proof:** Given a goal and using the strategy for building a tableau it is clear that it must be unique because there is no choice in which rule is to be applied to any subgoal. Moreover if a BAL can be applied in more than one way we always choose the premise with least distance from the current goal. The main part of the proof is to show that any tableau for a starting goal  $\alpha \doteq \beta$  has a bounded size.

$M$  is the maximum norm associated with the grammar. The size of a goal  $\gamma \doteq \delta$  in a tableau is  $\max\{|\gamma|, |\delta|\}$ . Now we make a number of observations which prove the result. The first observation is straightforward.

- (1) For any  $m \geq 0$  there are only boundedly many different goals whose size is at most  $m$ .

The next observation places a bound on the size of any subgoal which is the result of a BAL followed by a CUT.

- (2) After an application of BAL followed by CUT the size of the resulting subgoal is no more than  $3M + 1$ .

This property follows from the observations made in the previous section, when discussing balance. Hence any branch of a tableau contains only boundedly many applications of BAL, because otherwise goals will be repeated and a repeated goal is final.

A goal is small if its size is less than  $(M^2 + M)$ . Clearly there are only boundedly many small goals. Hence any branch of a tableau involving only small goals and only applications of UNF must be boundedly finite, as it must eventually contain a successful final goal.

If a goal is not small, we say it is “large”. The final property is to show that any branch in a tableau where there are only applications of UNF and no applications of BAL, but which contains large goals must be declining in size. More precisely, assume a large goal  $\alpha \doteq \beta$  and without loss of generality assume that  $|\alpha| \geq |\beta|$ . The following property holds.

- (3) If there is a branch of the tableau starting from the large goal  $\alpha \doteq \beta$  which contains  $M^2 + M$  applications of UNF and no application of other rules, then there is a goal in this branch whose size is strictly smaller than that of  $\alpha \doteq \beta$ .

By assumption  $|\alpha| \geq |\beta|$  and as this goal is large  $\alpha = X_1 \dots X_n \alpha'$  where  $n \geq M$ , and  $|\alpha'| > 0$ . Therefore there is the following branch in the tableau.

$$\begin{array}{rcl}
 X_1 \dots X_n \alpha' & \doteq & \beta \\
 & \vdots & \\
 X_2 \dots X_n \alpha' & \doteq & \beta_1 \\
 & \vdots & \\
 & \vdots & \\
 X_n \alpha' & \doteq & \beta_{n+1} \\
 & \vdots & \\
 \alpha' & \doteq & \beta_{n+2}
 \end{array}$$

and there are at least  $M^2$  applications of UNF between the goal  $\alpha \doteq \beta$  and  $\alpha' \doteq \beta_{n+2}$ . Because neither BAL is applicable, it follows that there are at most  $M$  applications of UNF between the goals  $X_i \dots X_n \alpha' \doteq \beta_{i+1}$  and  $X_{i+1} \dots X_n \alpha' \doteq \beta_{i+2}$  (because otherwise BAL(L) would be applied). If any of the goals  $X_i \dots X_n \doteq \beta_{i+1}$  has smaller size than  $\alpha \doteq \beta$  then the proof is finished. So assume that this is not the case. Therefore, in particular the size of  $X_2 \dots X_n \alpha' \doteq \beta_1$  has size at least that of  $\alpha \doteq \beta$ , and so  $|\beta_1| \geq |\alpha|$ . Consequently  $|\beta_1| > |\beta|$ , and moreover  $|\beta| \geq (|\alpha| + 1) - M$ , that is at least  $M^2 + 1$ , because otherwise BAL(R) would apply. So  $\beta$  has the form  $Y_1 \dots Y_M \beta'$  where  $|\beta'| > 0$ . Because BAL(R) does not apply it follows that within  $M^2$  applications of UNF from  $\alpha \doteq \beta$  one of the goals has the form  $\gamma X_{i+1} \dots X_n \doteq \beta'$ . This goal must occur as follows for some  $i \leq n$ .

$$\begin{array}{rcl}
 X_i \dots X_n \alpha' & \doteq & \beta_{i+1} \\
 & \vdots & \\
 \gamma X_{i+1} \dots X_n \alpha' & \doteq & \beta' \\
 & \vdots & \\
 X_{i+1} \dots X_n \alpha' & \doteq & \beta_{i+2}
 \end{array}$$

By assumption the size of the goal  $X_{i+1} \dots X_n \alpha' \doteq \beta_{i+2}$  is at least the size of  $\alpha \doteq \beta$ , but this is impossible because  $|\beta_{i+2}| < |\beta'| + M$ .  $\square$

**Theorem 2 [Soundness]** *If the tableau for  $\alpha \doteq \beta$  is successful then  $\alpha \sim \beta$ .*

**Proof:** Suppose there is a successful tableau for  $\alpha \doteq \beta$  but  $\alpha \not\sim \beta$ . By theorem 1 this tableau is finite. By proposition 4 of section 2.3 there is a least approximant  $n$  such that  $\alpha \not\sim_n \beta$ . We construct an offending path of false goals through the tableau within which the approximant indices decrease whenever UNF is applied (by fact 2 of section 2.3). The other rules (BAL and CUT) preserve the falsity indices as we noted in fact 4 and proposition 4 of section 2.3. Because the tableau is finite and successful this means that the path of false goals must conclude with a final goal. But this is impossible. Clearly it is not possible to reach a final goal of the form  $\gamma \doteq \gamma$  with  $\gamma \not\sim_k \gamma$ . Moreover it is not possible to reach a final goal which is a repeat. At the first instance there is a least  $k$  such that  $\gamma \not\sim_k \delta$  and as there is at least one application between the goals this would imply that  $\gamma \not\sim_{k-1} \delta$ , which contradicts that  $k$  is the least approximant such that  $\gamma \not\sim_k \delta$ .  $\square$

**Theorem 3 [Completeness]** *If  $\alpha \sim \beta$  then the tableau for  $\alpha \doteq \beta$  is successful.*

**Proof:** One just builds the tableau for  $\alpha \doteq \beta$ . By facts 1 and 3 and proposition 3 of section 2.3 the application of any rule preserves truth. Therefore it is not possible to reach an unsuccessful final goal, and by theorem 1 above the tableau for  $\alpha \doteq \beta$  is finite, and therefore successful.  $\square$

This is by no means an optimal decision procedure for simple grammars. There is a much more elegant proof using *unique prime decomposition* of a configuration, for details see [6].

### 3 A Deeper Normal Form for DPDA

We show that (slightly extended) PDGs provide a normal form for DPDA.

#### 3.1 Congruence and cancellation

What are the key features that underpin decidability of language equivalence for simple grammars? First, language (or bisimulation) equivalence is a *congruence* with respect to stack prefixing (proposition 1 of section 2.3). Second, language (or bisimulation) equivalence supports *cancellation* of postfix stacks (proposition 3 of section 2.3).

- if  $L(\alpha) = L(\beta)$  then  $L(\delta\alpha) = L(\delta\beta)$
- if  $L(\alpha\delta) = L(\beta\delta)$  then  $L(\alpha) = L(\beta)$

Congruence allows us to tear apart a configuration  $\alpha'\alpha$  and replace its tail with a potentially equivalent configuration  $\beta$  with overall result  $\alpha'\beta$ , as used in the BAL rules. Cancellation, as used in CUT, has the consequence that goals have bounded size. (Soundness of congruence and cancellation depend on dual versions that employ approximants.)

We might try to extend the tableau proof rules to goals that involve DPDA configurations. The rule UNF works for *stable* configurations.

$$\frac{p\alpha \doteq q\beta}{(p\alpha \cdot a_1) \doteq (q\beta \cdot a_1), \dots, (p\alpha \cdot a_k) \doteq (q\beta \cdot a_k)}$$

where  $(r\delta \cdot a)$  is the unique stable configuration  $r'\delta'$  such that  $r\delta \xrightarrow{a} r'\delta'$  is an edge in  $G^c(r\delta)$  or  $\emptyset$ . The problem is the other rules. How can we tear apart DPDA (stable) configurations and replace parts with parts? Indeed, what is a *part* of a configuration?

The problem is that a configuration has a state as well as a stack. Clearly,  $L(p\alpha) = L(q\beta)$  does not, in general, imply  $L(p\delta\alpha) = L(q\delta\beta)$ . For instance, with the following four transitions,  $L(pB) = L(qC)$  and  $L(pAB) \neq L(qAC)$ .

$$pA \xrightarrow{a} p\epsilon \quad qA \xrightarrow{b} q\epsilon \quad pB \xrightarrow{c} p\epsilon \quad qC \xrightarrow{c} q\epsilon$$

Also,  $L(p\alpha\delta) = L(q\beta\delta)$  does not, in general, imply  $L(p\alpha) = L(q\beta)$ . Consider the DPDA example 4 of section 1.1. The collapsed graph  $G^c(pYX)$  is pictured in figure 10. Although  $L(pY^nX) = L(pY^mX)$  for every  $m$  and  $n$ ,  $L(pY^n) = L(pY^m)$  only if  $n = m$ .

This example also illustrates that there is not an obvious relation between the lengths of stacks of equivalent configurations. A main attack on the decision problem in the 1970s examined differences between stack lengths and potentially equivalent configurations that eventually resulted in a proof of decidability for real-time DPDAs, [22]. (Indeed, this is a key insight for proving decidability of language containment for visibly pushdown automata, see section 1.4: if  $L(p\alpha) \subseteq L(q\beta)$  then  $|\alpha| = |\beta|$ .)

The problem with stack length is also apparent in the UNF rule for stable configurations. The configuration  $pY^nX$  has stack length  $n + 1$ , but  $(pY^nX \cdot b)$  has stack length 0. In the case of simple grammars, there is a tight bound on length of stacks after applications UNF.

Many of these problems disappear with pushdown grammars. Despite nondeterminism, stacking is a congruence with respect to pre and postfixing for language

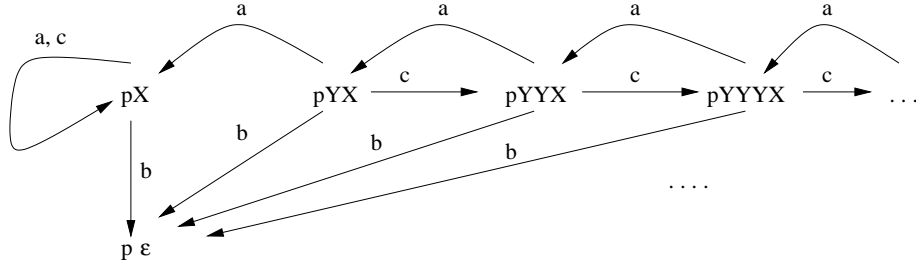


Figure 10: The graph  $G^c(pYX)$

and bisimulation equivalence and both equivalences support pre and postfix cancellation.

**Fact 1**

1.  $L(\alpha) = L(\beta)$  iff  $L(\alpha\delta) = L(\beta\delta)$
2.  $L(\alpha) = L(\beta)$  iff  $L(\delta\alpha) = L(\delta\beta)$
3.  $\alpha \sim \beta$  iff  $\alpha\delta \sim \beta\delta$
4.  $\alpha \sim \beta$  iff  $\alpha\delta \sim \beta\delta$

This suggests that we should try to remain with PDGs. Deterministic PDGs, as we have seen, are too restricted. In contrast, arbitrary PDGs are too rich (as they generate all the non-empty context-free languages). What is needed is a mechanism for constraining nondeterminism in a PDG.

### 3.2 Strict deterministic grammars

In this section strict deterministic grammars are described. They were introduced by Harrison and Havel [15], and further studied in [14, 16]. They generate exactly the same languages as DPDA with empty stack acceptance.

A SDG, strict deterministic grammar, is a PDG with an extra component.

- An equivalence relation  $\equiv$  on  $S$

The relation  $\equiv$  partitions the stack symbols  $S$  into disjoint subsets  $S_1, \dots, S_k$ : for each  $i$ , and pair of stack symbols  $X, Y \in S_i$ ,  $X \equiv Y$ . We extend  $\equiv$  on  $S$  to a relation on  $S^*$  using the same notation.

**Definition 1**  $\alpha \equiv \beta$  iff

1.  $\alpha = \beta$ , or
2. there is a  $\delta \in S^*$ ,  $\alpha = \delta X \alpha'$ ,  $\beta = \delta Y \beta'$ ,  $X \equiv Y$  and  $X \neq Y$ .

Consequently,  $\alpha \equiv \beta$  if they are the same or if they have a common prefix followed by different stack symbols belonging to the same equivalence class. For instance, if  $Y \equiv Z$  then  $XY\alpha \equiv XZ$  for any  $\alpha$ . (The relation  $\equiv$  on stacks is *not* an equivalence relation.)

**Fact 1**

1.  $\alpha\beta \equiv \alpha$  iff  $\beta = \epsilon$
2.  $\alpha \equiv \beta$  iff  $\delta\alpha \equiv \delta\beta$

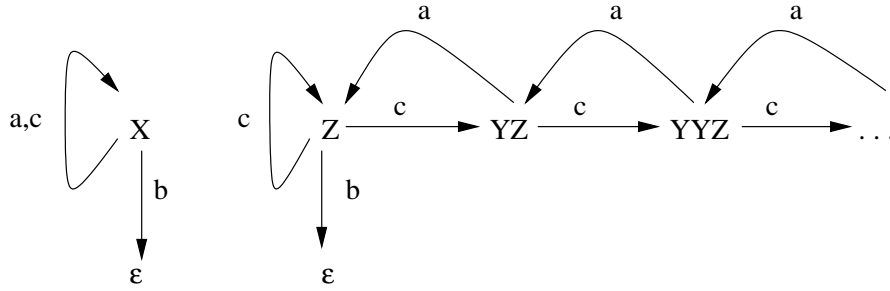


Figure 11: The graph  $G(X)$  and  $G(Z)$

3. If  $\alpha \equiv \beta$  and  $\gamma \equiv \delta$  then  $\alpha\gamma \equiv \beta\delta$
4. If  $\alpha \equiv \beta$  and  $\alpha \neq \beta$  then  $\alpha\gamma \equiv \beta\delta$
5. If  $\alpha\gamma \equiv \beta\delta$  and  $|\alpha| = |\beta|$  then  $\alpha \equiv \beta$

**Definition 2** The relation  $\equiv$  on  $S$  is strict if the following two conditions hold

1. if  $\begin{array}{c} X \xrightarrow{a} \alpha \in T \\ ||| \\ Y \xrightarrow{a} \beta \in T \end{array}$  then  $\begin{array}{c} \alpha \\ ||| \\ \beta \end{array}$
2. if  $\begin{array}{c} X \xrightarrow{a} \alpha \in T \\ ||| \\ Y \xrightarrow{a} \alpha \in T \end{array}$  then  $\begin{array}{c} X \\ || \\ Y \end{array}$

**Definition 3** An enhanced PDG is a *strict deterministic grammar*, SDG, if its relation  $\equiv$  is strict.

**Example 1**  $S = \{X, Y, Z\}$ ,  $A = \{a, b, c\}$  and  $T$  is

$$\begin{array}{cccc} X \xrightarrow{a} X & X \xrightarrow{b} \epsilon & X \xrightarrow{c} X & Y \xrightarrow{a} \epsilon \\ Y \xrightarrow{c} YY & Z \xrightarrow{b} \epsilon & Z \xrightarrow{c} Z & Z \xrightarrow{c} YZ \end{array}$$

The graphs of  $X$  and  $Z$  are pictured in figure 11.  $G(Z)$  is nondeterministic because there are two  $c$ -transitions from  $Z$ . Consider the following partition of  $S$ :  $\{\{X\}, \{Y, Z\}\}$ . This enhanced PDG is an SDG. An examination of its transitions reveals that only condition 1 of strictness needs to be checked for each pair of the three transitions  $Y \xrightarrow{c} YY$ ,  $Z \xrightarrow{c} Z$  and  $Z \xrightarrow{c} YZ$ . However,  $YY \equiv Z$ ,  $YY \equiv YZ$  and  $Z \equiv YZ$  because  $Y$  and  $Z$  belong to the same partition.  $\square$

**Example 2** A second SDG example is from [15]. The set  $S$  is  $\{S, A, B, C, D, A', C'\}$  and its partition is  $\{\{S\}, \{A, B\}, \{C, D\}, \{A'\}, \{C'\}\}$ ,  $A = \{a, b, c\}$  and  $T$  is

$$\begin{array}{cccc} S \xrightarrow{a} A & S \xrightarrow{a} B & A \xrightarrow{a} AA' & A \xrightarrow{b} C \\ B \xrightarrow{a} B & B \xrightarrow{b} D & C \xrightarrow{a} \epsilon & C \xrightarrow{b} C' \\ D \xrightarrow{b} DC' & D \xrightarrow{c} \epsilon & A' \xrightarrow{a} \epsilon & C' \xrightarrow{c} \epsilon \end{array}$$

The graph  $G(S)$  is pictured in figure 12.  $\square$

We now examine some features of an SDG. If each set in the partition is a singleton, so only  $X \equiv X$  for each  $X$ , then the SDG is *deterministic*, and is therefore a simple grammar. In this extreme case  $\alpha \equiv \beta$  iff  $\alpha = \beta$ . Examples 1 and 2

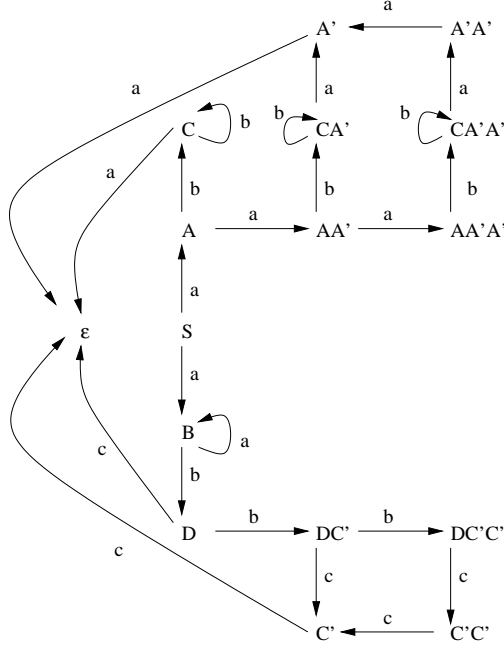


Figure 12: The graph  $G(S)$

illustrate that when the partition involves larger sets then the SDG may exhibit nondeterminism. However, nondeterminism is “constrained”. For example, the following holds.

- if  $X \xrightarrow{a} \epsilon \in \mathbb{T}$  and  $X \equiv Y$  and  $X \neq Y$ , then there is not an  $a$ -transition  $Y \xrightarrow{a} \beta \in \mathbb{T}$

To prove this, assume  $X \xrightarrow{a} \epsilon$  and  $X \equiv Y$  and  $X \neq Y$  and  $Y \xrightarrow{a} \beta$ . By condition 1 of being strict  $\epsilon \equiv \beta$ . But then  $\beta = \epsilon$ . And by condition 2 of being strict  $X = Y$ , which is a contradiction. The next two properties show that the strictness conditions generalise to all words  $w \in \mathbf{A}^*$ .

- if  $X \xrightarrow{w} \alpha$  and  $Y \xrightarrow{w} \beta$  and  $X \equiv Y$  then  $\alpha \equiv \beta$
- if  $X \xrightarrow{w} \alpha$  and  $Y \xrightarrow{w} \alpha$  and  $X \equiv Y$  then  $X = Y$

To prove these we show an even more general result.

**Proposition 1**

1. If  $\alpha \xrightarrow{w} \alpha'$  and  $\beta \xrightarrow{w} \beta'$  and  $\alpha \equiv \beta$  then  $\alpha' \equiv \beta'$ .
2. If  $\alpha \xrightarrow{w} \alpha'$  and  $\beta \xrightarrow{w} \alpha'$  and  $\alpha \equiv \beta$  then  $\alpha = \beta$ .

**Proof:** In both cases the proof is by induction on  $|w|$ . For the base case of 1  $|w| = 0$ . In which case  $\alpha \xrightarrow{w} \alpha$  and  $\beta \xrightarrow{w} \beta$  and by assumption  $\alpha \equiv \beta$ . For the inductive step, assume  $w = aw'$  and  $\alpha \xrightarrow{a} \alpha_1 \xrightarrow{w'} \alpha'$  and  $\beta \xrightarrow{a} \beta_1 \xrightarrow{w'} \beta'$ . Therefore  $\alpha = X\delta$  and  $X \xrightarrow{a} \delta_1$  and  $\alpha_1 = \delta_1\delta$  and  $\beta = Y\gamma$  and  $Y \xrightarrow{a} \gamma_1$  and  $\beta_1 = \gamma_1\gamma$ . That is we have the following.

$$\begin{array}{ccc}
 X\delta & & Y\gamma \\
 \downarrow a & & \downarrow a \\
 \alpha_1 = \delta_1\delta & & \gamma_1\gamma = \beta_1 \\
 \downarrow w' & & \downarrow w' \\
 \alpha' & & \beta'
 \end{array}$$

Because  $X\delta \equiv Y\gamma$  it follows that  $X \equiv Y$  and therefore  $\delta_1 \equiv \gamma_1$  by the first condition of the definition of strictness. There are two cases to consider.

**Case I:**  $\delta_1 = \gamma_1$ , and therefore by condition 2 of being strict  $X = Y$  and therefore because  $X\delta \equiv Y\gamma$  it follows that  $\delta \equiv \gamma$  and therefore  $\delta_1\delta \equiv \gamma_1\gamma$ . Now the required result,  $\alpha' \equiv \beta'$ , follows by the induction hypothesis because  $\alpha_1 \equiv \beta_1$  and  $|w'| < |w|$ .

**Case II:**  $\delta_1 \neq \gamma_1$ , and therefore because  $\delta_1 \equiv \gamma_1$  it now follows that  $\delta_1\delta \equiv \gamma_1\gamma$ . The required result now follows as in case I.

The base case for 2 is again when  $|w| = 0$ . Therefore  $\alpha' = \alpha$  and  $\alpha' = \beta$  and therefore  $\alpha = \beta$ . For the inductive step assume  $w = aw'$  and  $\alpha \xrightarrow{a} \alpha_1 \xrightarrow{w'} \alpha'$  and  $\beta \xrightarrow{a} \beta_1 \xrightarrow{w'} \alpha'$ . As in the case of the proof of 1,  $\alpha = X\delta$  and  $X \xrightarrow{a} \delta_1$  and  $\alpha_1 = \delta_1\delta$  and  $\beta = Y\gamma$  and  $Y \xrightarrow{a} \gamma_1$  and  $\beta_1\gamma_1\gamma$ , and so we have the following.

$$\begin{array}{ccc} X\delta & & Y\gamma \\ \downarrow a & & \downarrow a \\ \alpha_1 = \delta_1\delta & & \gamma_1\gamma = \beta_1 \\ \downarrow w' & & \downarrow w' \\ \alpha' & & \alpha' \end{array}$$

We can use the same argument as above to show that  $\alpha_1 \equiv \beta_1$  and therefore by the induction hypothesis because  $\alpha_1 \xrightarrow{w'} \alpha'$  and  $\beta_1 \xrightarrow{w'} \alpha'$  it follows that  $\alpha_1 = \beta_1$ . Therefore  $\delta_1\delta = \gamma_1\gamma$ . However  $\delta_1 \equiv \gamma_1$ . By the definition of  $\equiv$  it is not possible for any sequence  $\alpha \equiv \alpha X\lambda$ . Therefore  $\delta_1 = \gamma_1$  and  $\delta = \gamma$ . But also  $X \xrightarrow{a} \delta_1$  and  $Y \xrightarrow{a} \gamma_1$  and so by the second condition of being strict  $X = Y$ . The result,  $X\delta = Y\gamma$ , is now proved.  $\square$

If  $\alpha \equiv \beta$  then their languages are prefix disjoint (part 1 below) and if  $\alpha \equiv \beta$  and  $\alpha \neq \beta$  then their languages are disjoint (part 2 below).

### Proposition 2

1. If  $\alpha \equiv \beta$  and  $w \in L(\alpha)$ , then for all words  $v$ , and  $a \in \mathbf{A}$ ,  $wav \notin L(\beta)$
2. If  $\alpha \equiv \beta$  and  $\alpha \neq \beta$  then  $L(\alpha) \cap L(\beta) = \emptyset$

**Proof:** To show 1 assume  $\alpha \xrightarrow{w} \epsilon$  and  $\alpha \equiv \beta$ . If  $\beta \xrightarrow{w} \gamma$  then by proposition 1.1  $\epsilon = \gamma$ , and so  $\alpha = \beta$ . Therefore, it is not possible that  $\beta \xrightarrow{wav} \epsilon$ . Part 2 is now a corollary.  $\square$

### 3.3 Sum configurations

In this section we move towards an alternative characterisation of the collapsed graphs of DPDA using SDG. The characterisation of the collapsed graphs is not up to isomorphism, because, for instance, all terminal vertices  $pe$  of a collapsed graph are identified. However it does preserve bisimulation equivalence. The key construction is *determinization* of SDG.

A configuration of a SDG is a sequence of stack symbols,  $\alpha$ . We extend the notion of a configuration to sets of sequences of stack symbols,  $\{\alpha_1, \dots, \alpha_n\}$ . However, we shall write a set as a sum,  $\alpha_1 + \dots + \alpha_n$  without repeated elements. Two sum configurations  $\alpha_1 + \dots + \alpha_n$ , and  $\beta_1 + \dots + \beta_m$  are identical if they are the same set,  $\{\alpha_1, \dots, \alpha_n\} = \{\beta_1, \dots, \beta_m\}$ . Therefore, we identify sum configurations which are permutations of each other<sup>6</sup>. A degenerate case is the empty sum, which we write

<sup>6</sup>If we were to assume a total ordering on  $\mathbf{S}$ , then there would be a unique sum form  $\alpha_1 + \dots + \alpha_n$  where  $\alpha_i$  is smaller than  $\alpha_j$ ,  $i < j$ , with respect to lexicographical ordering.

as  $\emptyset$ . The language of a sum configuration is just the union of the languages of the components.

$$L(\alpha_1 + \dots + \alpha_n) = \bigcup \{L(\alpha_i) : 1 \leq i \leq n\}$$

Therefore,  $L(\emptyset) = \emptyset$ .

**Example 1** Consider the sum configuration  $X + Z$  of example 1 of the previous section.  $L(X + Z)$  is the set of words  $\{a^n b : n \geq 0\} \cup \{c^n c^m a^m b : n, m \geq 0\}$ .  $\square$

Only a subset of sum configurations are interesting.

**Definition 1**  $\beta_1 + \dots + \beta_n$  is *admissible*, if  $\beta_i \equiv \beta_j$  for each  $i$  and  $j$ .

The empty sum,  $\emptyset$ , is therefore admissible. In [16] admissible configurations are called “associates”.

**Example 2** The following are admissible configurations from example 1 of the previous section:  $XX$ ,  $ZZZ + ZZY$ ,  $YX + Z$ ,  $Z + YZ$ ,  $Z + YZ + YYZ$ . The last example is admissible because  $Z \equiv YZ$ ,  $Z \equiv YYZ$  and  $YZ \equiv YYZ$ . The sum  $X + Z$  is *not* admissible.  $\square$

An important property of admissible configurations is that they are preserved by transitions (which follows from proposition 1 of the previous section).

**Fact 1** If  $\alpha_1 + \dots + \alpha_n$  is admissible then for all words  $w$  the sum configuration

$$\{\beta_i^1 : \alpha_1 \xrightarrow{w} \beta_i^1\} \cup \dots \cup \{\beta_i^n : \alpha_n \xrightarrow{w} \beta_i^n\}$$

is admissible.

An application is *determinization* of a SDG where  $\mathbb{T}$  is changed to  $\mathbb{T}^d$ . For each stack symbol  $X$  and  $a \in \mathbb{A}$ , the family of transitions

$$X \xrightarrow{a} \alpha_1, \dots, X \xrightarrow{a} \alpha_n \in \mathbb{T}$$

is determinized by replacing them with the single transition

$$X \xrightarrow{a} \alpha_1 + \dots + \alpha_n \in \mathbb{T}^d$$

By definition, the sum configuration  $\alpha_1 + \dots + \alpha_n$  is admissible. Note that for each stack symbol  $X$  and  $a \in \mathbb{A}$  there is such a unique transition  $X \xrightarrow{a} \sum \alpha_i \in \mathbb{T}^d$ : if there is not an  $a$  transition for  $X$  in  $\mathbb{T}$  then  $X \xrightarrow{a} \emptyset \in \mathbb{T}^d$ . Next, the prefix rule PRE for generating transitions is generalised to cover admissible configurations.

$$\begin{array}{l} \text{PRE} \quad \text{If } X_i \xrightarrow{a} \alpha_1^i + \dots + \alpha_{n_i}^i \quad \text{for each } i : 1 \leq i \leq m \text{ then} \\ X_1 \beta_1 + \dots + X_m \beta_m \xrightarrow{a} \alpha_1^1 \beta_1 + \dots + \alpha_{n_1}^1 \beta_1 \quad + \\ \quad + \\ \quad + \\ \quad \alpha_1^m \beta_m + \dots + \alpha_{n_m}^m \beta_m \end{array}$$

If the antecedent sum configuration of PRE is admissible then so is the consequent configuration. The determinized transition graph  $\mathbb{G}^d(\alpha_1 + \dots + \alpha_n)$  is generated from the admissible configuration  $\alpha_1 + \dots + \alpha_n$  using the determinized transitions  $\mathbb{T}^d$  and the extended prefix rule.

**Example 3** We reconsider example 1 of section 3.2, where  $\mathbb{S} = \{X, Y, Z\}$ ,  $\mathbb{A} = \{a, b, c\}$  and the partition of  $\mathbb{S} = \{\{X\}, \{Y, Z\}\}$ . Below is  $\mathbb{T}^d$ .

$$\begin{array}{lll} X \xrightarrow{a} X & Y \xrightarrow{a} \epsilon & Z \xrightarrow{a} \emptyset \\ X \xrightarrow{b} \epsilon & Y \xrightarrow{b} \emptyset & Z \xrightarrow{b} \epsilon \\ X \xrightarrow{c} X & Y \xrightarrow{c} YY & Z \xrightarrow{c} YZ + Z \end{array}$$

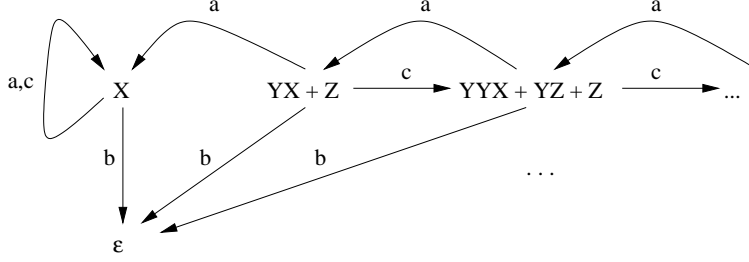


Figure 13: The graph  $G^d(YX + Z)$

The graph  $G^d(YX + Z)$  is pictured in figure 13. It is not an accident that this graph is isomorphic to the graph of figure 10, as we shall see later. We examine some of the transitions of figure 13, and how they are derived using PRE.

1.  $YX + Z \xrightarrow{a} X$  because  $YX \xrightarrow{a} X$  and  $Z \xrightarrow{a} \emptyset$ .
2.  $YX + Z \xrightarrow{b} \epsilon$  because  $YX \xrightarrow{b} \emptyset$  and  $Z \xrightarrow{b} \epsilon$ .
3.  $YX + Z \xrightarrow{c} YYX + YZ + Z$  because  $YX \xrightarrow{c} YYX$  and  $Z \xrightarrow{c} YZ + Z$ .

All sum configurations in the graph are admissible.  $\square$

### 3.4 Characterising DPDA

A DPDA can be converted into a SDG, and configuration  $p\alpha$  is converted into the *sum* configuration  $\text{sum}(p\alpha)$  of the SDG where  $G^c(p\alpha) \sim G^d(\text{sum}(p\alpha))$ .

There is a standard transformation of a pushdown automaton into a language equivalent context-free grammar: see, for instance, [14, 17]. Assume a DPDA in normal form with components  $P, S, A$  and  $T$ . A SDG in normal form with components  $S_1, A, T_1$  and  $\equiv$  is constructed, in stages.

1. For every  $p, q \in P$  and  $X \in S$ , introduce a stack symbol  $[pXq] \in S_1$ .
2. For transitions, the initial step is to define the following transitions for  $a \in A$ .
  - (a) If  $pX \xrightarrow{a} q\epsilon \in T$ , then  $[pXq] \xrightarrow{a} \epsilon \in T_1$ .
  - (b) If  $pX \xrightarrow{a} qY \in T$ , then  $[pXr] \xrightarrow{a} [qYr] \in T_1$  for each  $r \in P$ .
  - (c) If  $pX \xrightarrow{a} qYZ \in T$ , then  $[pXr] \xrightarrow{a} [qYp'][p'Zr] \in T_1$  for each  $r, p' \in P$ .
3. A stack symbol  $[pXq] \in S_1$  is an  $\epsilon$ -symbol, if  $pX \xrightarrow{\epsilon} q\epsilon \in T$ . All  $\epsilon$ -symbols are erased from the right hand side of any transition in  $T_1$ .
4. Next, the SDG is normalised by removing all unnormed stack symbols and useless transitions (as in section 2.1).
5. Finally, for elements of  $S_1$ , if  $p = r$  then  $[pSq] \equiv [rSt]$  for any  $q, t$ .

The following captures features of the SDG that is built from the DPDA.

**Fact 1**

1. If  $w \in A^*$  and  $pX$  is stable in the DPDA, then  $pX \xrightarrow{w} q\epsilon$  if, and only if,  $[pXq] \xrightarrow{w} \epsilon$  in the SDG.
2. The relation  $\equiv$  on  $S_1$  is an equivalence relation and is strict.

Harrison and Havel also prove the converse, that any strict SDG can be transformed into a DPDA [15].

The transformation of a DPDA into a strict SDG does not preserve determinism. However, if the SDG is determinized, then it is preserved. Moreover, any configuration  $pX_1X_2 \dots X_n$  of the DPDA is transformed into the admissible configuration  $\text{sum}(p\alpha) = \sum [pX_1p_1][p_1X_2p_2] \dots [p_{n-1}X_n p_n]$  where the summation is over all  $p_i \in \mathbf{P}$ , for  $1 \leq i \leq n$  after all  $\epsilon$ -symbols are erased, and summands involving redundant stack symbols are removed.

**Fact 2**  $L(p\alpha) = L(\text{sum}(p\alpha))$

Moreover, the transition graph  $\mathbf{G}^c(p\alpha)$  is almost isomorphic to<sup>7</sup>  $\mathbf{G}^d(\text{sum}(p\alpha))$ . There can be  $|\mathbf{P}|$  co-roots, vertices of the form  $q\epsilon$ , in  $\mathbf{G}^c(p\alpha)$  in contrast with the single co-root,  $\epsilon$ , in  $\mathbf{G}^d(\text{sum}(p\alpha))$ .

**Example 1** An example is the conversion of the DPDA of example 4 of the section 1.1. The stack symbols  $\mathbf{S}_1$  is the set

$$\{[pXp], [pXr], [pYp], [pYr], [rXp], [rXr], [rYp], [rYr]\}$$

The transitions in  $\mathbf{T}$  are then converted.

$$\begin{array}{lll} [pXp] \xrightarrow{a} [pXp] & [pXr] \xrightarrow{a} [pXr] & [pXp] \xrightarrow{b} \epsilon \\ [pXp] \xrightarrow{c} [pXp] & [pXr] \xrightarrow{c} [pXr] & \\ [pYp] \xrightarrow{a} \epsilon & [pYr] \xrightarrow{b} \epsilon & [pYp] \xrightarrow{c} [pYp][pYp] \\ [pYp] \xrightarrow{c} [pYr][rYp] & [pYr] \xrightarrow{c} [pYp][pYr] & [pYr] \xrightarrow{c} [pYr][rYr] \end{array}$$

There are two  $\epsilon$ -stack symbols,  $[rXp]$  and  $[rYr]$ . These are erased from the right hand side of any transition: the transition  $[pYr] \xrightarrow{c} [pYr][rYr]$  is changed to  $[pYr] \xrightarrow{c} [pYr]$ . The stack symbols  $[pXr]$ ,  $[rYp]$ ,  $[rXp]$ ,  $[rXr]$  and  $[rYr]$  are redundant and are removed. This reduces  $\mathbf{S}_1$  to the set  $\{[pXp], [pYp], [pYr]\}$ , and the transitions  $\mathbf{T}_1$  to the following set

$$\begin{array}{lll} [pXp] \xrightarrow{a} [pXp] & [pXp] \xrightarrow{b} \epsilon & [pXp] \xrightarrow{c} [pXp] \\ [pYp] \xrightarrow{a} \epsilon & [pYr] \xrightarrow{b} \epsilon & [pYp] \xrightarrow{c} [pYp][pYp] \\ [pYr] \xrightarrow{c} [pYp][pYr] & [pYr] \xrightarrow{c} [pYr] & \end{array}$$

The partition is into the sets  $\{\{[pXp]\}, \{[pYp], [pYr]\}\}$ . Finally, the transitions  $\mathbf{T}_1$  are determinised: the two  $c$ -transitions from  $[pYr]$  are replaced by the single transition  $[pYr] \xrightarrow{c} [pYr] + [pYp][pYr]$ . The resulting SDG is example 3 of the previous section when  $X = [pXp]$ ,  $Y = [pYp]$  and  $Z = [pYr]$ . The configuration  $pY Y X$  of the DPDA becomes the admissible configuration  $[pYp][pYp][pXp] + [pYp][pYr] + [pYr]$  of the SDG, that generates the same language.  $\square$

The DPDA decision problem is, therefore, equivalent to deciding whether two admissible configurations of a determinized SDG are bisimulation equivalent. A DPDA in normal form with stack size  $s$  and state size  $p$  is transformed into a SDG whose stack size at most  $s \times p^2$  and where the size of each element of the partition of the stack symbols is at most  $p$ . Two configurations  $p\alpha$  and  $q\beta$  of a DPDA in normal form, whose stack lengths are at most  $n$ , become admissible configurations of the strict SDG that contain stack sequences whose length is bounded by  $n$ .

<sup>7</sup>In fact, is bisimulation equivalent to.

## 4 Decidability of DPDA Equivalence

The DPDA equivalence problem reduces to deciding whether two admissible configurations of a determinized SDG are bisimulation equivalent. The problem is a natural generalisation of the decision question for simple grammars,  $\alpha \sim \beta?$ , that includes summation  $\alpha_1 + \dots + \alpha_n \sim \beta_1 + \dots + \beta_m?$

### 4.1 Heads, tails and extensions

Let  $E, F, G, \dots$  range over admissible configurations.

**Definition 1** The *size* of an admissible configuration  $E = \beta_1 + \dots + \beta_n$ , written  $|E|$ , is the length of its longest sequence,  $\max\{|\beta_j| : 1 \leq j \leq n\}$ . And  $|\emptyset| = 0$ .

For each  $n \geq 0$ , there are only boundedly many different admissible configurations of size at most  $n$ .

An admissible configuration is written  $\beta_1 + \dots + \beta_n$  where each  $\beta_i$  is distinct. The operator  $+$  can be extended: if  $E$  and  $F$  are admissible,  $E \cup F$  is admissible,  $E$  and  $F$  are disjoint,  $E \cap F = \emptyset$ , then  $E + F$  is the admissible configuration  $E \cup F$ . The operator  $+$  on admissible configurations is *partial*. We also use sequential composition, written as juxtaposition: if  $E$  and  $F$  are admissible, then  $EF$  is the admissible configuration  $\{\beta\gamma : \beta \in E \text{ and } \gamma \in F\}$ . The following are easy consequences of the properties of being admissible from section 3.3.

#### Fact 1

1. If  $E + F$  is admissible and  $u \in L(E)$ , then for any  $v$ ,  $uv \notin L(F)$ .
2. If  $E + F$  is admissible, then  $L(E) \cap L(F) = \emptyset$ .
3.  $L(EF) = \{uv : u \in L(E) \text{ and } v \in L(F)\}$ .

Admissible configurations can have different “shapes”, using  $+$  and sequential composition. If  $E = \{X'_1\gamma_1, \dots, X'_m\gamma_m\}$ , then for each  $i$  and  $j$ ,  $X_i \equiv X_j$ . Assume that the different stack symbols in  $\{X'_1, \dots, X'_m\}$  are  $X_1, \dots, X_n$ . Therefore,  $E = X_1G_1 + \dots + X_nG_n$  where each  $G_i = \{\gamma_j : X'_j = X_i\}$ . Clearly,  $X_1 + \dots + X_n$  is admissible and each  $G_i$  is admissible. In this presentation,  $E$  is in *1-head form*.

**Definition 2** Assume  $k \geq 1$  and  $E = \{\beta'_1\delta_1, \dots, \beta'_m\delta_m\}$  and  $|\beta'_i| = k$ , or  $|\beta'_i| < k$  and  $\delta_i = \varepsilon$ , for each  $i : 1 \leq i \leq m$ . If  $\beta_1, \dots, \beta_n$  are the distinct elements in  $\{\beta'_1, \dots, \beta'_m\}$  and  $H_l = \{\delta_j : \beta'_j = \beta_l\}$  for each  $l : 1 \leq l \leq n$ , then  $E = \beta_1H_1 + \dots + \beta_nH_n$  is in *k-head form*.

**Fact 2** If  $E = \beta_1G_1 + \dots + \beta_nG_n$  is in *k-head form*, then  $\beta_1 + \dots + \beta_n$  is admissible and each  $G_i$  is admissible and different from  $\emptyset$ .

Head forms are instances of a more general head/tail form.

**Definition 3**  $E = E_1G_1 + \dots + E_nG_n$  is in *head/tail form*, if the head  $E_1 + \dots + E_n$  is admissible and at least one  $E_i \neq \emptyset$ , and each tail  $G_i \neq \emptyset$ .

**Example 1**  $E = YYYX + YYZ + YZ + Z$  is an admissible configuration of example 3 of section 3.3. The partition of the stack symbols is  $\{\{X\}, \{Y, Z\}\}$ .  $E$  has 1-head form,  $YG_1 + ZG_2$ , where  $G_1 = YXX + YZ + Z$  and  $G_2 = \varepsilon$ . Also,  $E$  has 2-head form,  $YH_1 + YZH_2 + ZH_3$ , where  $H_1 = YX + Z$  and  $H_2 = H_3 = \varepsilon$ .  $E$  cannot be presented as  $YYG'_1 + YYG'_2 + YG'_3 + ZG'_4$ : this is not a valid head/tail form because the head  $YY + YY + Y + Z$  is not admissible ( $YY \not\equiv Y$ ) and it is not disjoint ( $YY + YY$  is not a proper sum).  $\square$

In the following, if a configuration  $E$  is written  $E_1G_1 + \dots + E_nG_n$ , then we assume that it fulfills the conditions of definition 3 of a head/tail form.

A key definition, used in the decision procedure later, is when one configuration *tail extends* another. Consider first configurations of a simple grammar  $\alpha_1 = \beta_1\gamma$ ,  $\alpha_2 = \beta_2\lambda\gamma$  and  $\alpha_3 = \beta_3\delta\lambda\gamma$ . Each  $\alpha_i$  has possibly differing heads  $\beta_i$  and differing tails. However,  $\alpha_2$ 's tail *extends*  $\alpha_1$ 's (with  $\lambda$ ) and  $\alpha_3$ 's extends  $\alpha_2$ 's with  $\delta$ . Therefore,  $\alpha_3$ 's tail extends  $\alpha_1$ 's tail with the composition of  $\lambda$  and  $\delta$ . We now extend the idea to admissible configurations.

**Definition 4** If  $E = E_1G_1 + \dots + E_nG_n$ ,  $F = F_1H_1 + \dots + F_mH_m$  then  $F$  in its head/tail form is a *tail extension* of  $E$  in its head/tail form provided that for each  $i : 1 \leq i \leq m$ ,  $H_i = K_1^iG_1 + \dots + K_n^iG_n$ . If  $F$  is a tail extension of  $E$ , then the associated extension  $e$  is the  $m$ -tuple  $(K_1^1 + \dots + K_n^1, \dots, K_1^m + \dots + K_n^m)$  without the  $G_i$ s, and the size of  $e$ , written  $|e|$ , is  $\max\{|K_j^i| : 1 \leq i \leq n \text{ and } 1 \leq j \leq m\}$  and the width of  $e$  is  $m$ , and  $F$  is said to extend  $E$  with  $e$ .

Extensions are matrices, written in a linear notation. An instance of extension is when the tails coincide. If  $E = E_1G_1 + \dots + E_nG_n$  and  $F = F_1G_1 + \dots + F_nG_n$ , then  $E$  extends  $F$  by  $e = (\varepsilon + \emptyset + \dots + \emptyset, \dots, \emptyset + \emptyset + \dots + \varepsilon)$ . This extension  $e$  is abbreviated to the identity  $(\varepsilon)$ . Extensions can be composed.

**Proposition 1** If  $E = E_1G_1 + \dots + E_lG_l$  and  $E' = E'_1G'_1 + \dots + E'_mG'_m$  and  $E'' = E''_1G''_1 + \dots + E''_nG''_n$  and  $E'$  extends  $E$  by  $e = (J_1^1 + \dots + J_l^1, \dots, J_1^m + \dots + J_l^m)$  and  $E''$  extends  $E'$  by  $f = (K_1^1 + \dots + K_m^1, \dots, K_1^n + \dots + K_m^n)$ , then  $E''$  extends  $E$  by  $ef = (H_1^1 + \dots + H_l^1, \dots, H_1^n + \dots + H_l^n)$  where  $H_j^i = K_1^iJ_j^1 + \dots + K_m^iJ_j^m$  and  $|ef| \leq |e| + |f|$ .

**Proof:** Assume that  $E''$  extends  $E'$  by  $f$  and  $E'$  extends  $E$  by  $e$ . Therefore,  $G''_i = K_1^iG'_1 + \dots + K_m^iG'_m$  and  $G'_k = J_1^kG_1 + \dots + J_l^kG_l$ . Consequently,  $G''_i = C_1 + \dots + C_m$  where  $C_t = K_t^iJ_1^tG_1 + \dots + K_t^iJ_l^tG_l$ . Reorganising the expression,  $G''_i = H_1G_1 + \dots + H_lG_l$  where  $H_t = K_1^iJ_t^1 + K_2^iJ_t^2 + \dots + K_m^iJ_t^m$ , as required. Clearly  $|ef| \leq |e| + |f|$ .  $\square$

We are especially interested in tail extensions when configurations have the same heads.

**Example 2** The following uses example 3 of section 3.3.

$$\begin{aligned} E &= YG_1 + ZG_2 & \text{where } G_1 &= X \text{ and } G_2 = \varepsilon \\ E' &= YG'_1 + ZG'_2 & \text{where } G'_1 &= YX + Z \text{ and } G'_2 = \varepsilon \\ E'' &= YG''_1 + ZG''_2 & \text{where } G''_1 &= YYX + YZ + Z \text{ and } G''_2 = \varepsilon \end{aligned}$$

$E'$  extends  $E$  by  $e = (Y + Z, \emptyset + \varepsilon)$  and  $E''$  extends  $E'$  by  $f = e = (Y + Z, \emptyset + \varepsilon)$ . Therefore,  $E''$  extends  $E$  by  $ef = (YY + (YZ + Z), \emptyset + \varepsilon)$ .  $\square$

## 4.2 Dynamic properties of head and tail forms

We adopt notation from sections 2.1 and 2.2.

- For each  $X \in S$ ,  $w(X)$  is the unique shortest word  $v \in A^+$  such that  $X \xrightarrow{v} \varepsilon$
- $w(E)$  is the unique shortest word for configuration  $E$
- $M$  is the maximum norm of the SDG
- $E$  after  $u$ , written  $E \cdot u$ , is the unique  $F$  such that  $E \xrightarrow{u} F$

**Fact 1**  $(\alpha_1 + \dots + \alpha_n) \cdot u = (\alpha_1 \cdot u) + \dots + (\alpha_n \cdot u)$

**Example 1**  $E = YYYX + YYZ + YZ + Z$  is from example 3 of section 3.3.

$$\begin{aligned}(E \cdot c) &= (YY \cdot c)H_1 + (YZ \cdot c)H_2 + (Z \cdot c)H_3 \\ &= YYYH_1 + YYZH_2 + (YZ + Z)H_3\end{aligned}$$

where  $H_1 = YX + Z$  and  $H_2 = H_3 = \epsilon$ . And  $(E \cdot w(Z)) = \epsilon$ .  $\square$

A feature of the decision procedure is repeating patterns within admissible configurations.

**Proposition 1** *Assume  $E = \beta_1G_1 + \dots + \beta_nG_n$  is in  $k$ -head form, and each  $\beta_j = X_1^j \dots X_{k_j}^j$ .*

1. *If  $(\beta_i \cdot u) = \epsilon$ , then for each  $j \neq i$ ,  $(\beta_j \cdot u) = \emptyset$  and  $(E \cdot u) = G_i$ .*
2. *If  $(\beta_i \cdot u) = X_m^i \dots X_{k_i}^i$ , then  $(E \cdot u) = E_1G_1 + \dots + E_nG_n$  where  $E_i = (\beta_i \cdot u)$  and for  $j \neq i$ , either  $E_j = \emptyset$  or  $E_j = X_m^j \dots X_{k_j}^j$  and  $X_1^j \dots X_{m-1}^j$  is the same sequence as  $X_1^i \dots X_{m-1}^i$ .*

**Proof:** Part 1 follows from the definition of admissibility.  $\beta_1 + \dots + \beta_n$  is admissible and  $\beta_i \neq \beta_j$  when  $i \neq j$ . If  $(\beta_i \cdot u) = \epsilon$  and  $\beta_j \xrightarrow{u} F$ , then  $F \equiv \epsilon$  and, therefore,  $\beta_i = \beta_j$ . Therefore,  $(\beta_j \cdot u) = \emptyset$  and  $(E \cdot u) = (\beta_i \cdot u)G_i = G_i$ . Similar observations establish part 2.  $\square$

Bisimulation equivalence and its approximants are congruences with respect to + and sequential composition. In particular, head/tail forms allow substitutivity of equivalent subexpressions into tails (because admissibility is preserved).

**Proposition 2** *Assume  $E = E_1G_1 + \dots + E_nG_n$ .*

1. *If  $(E_i \cdot u) = \epsilon$ , then for all  $j \neq i$ ,  $(E_j \cdot u) = \emptyset$  and  $(E \cdot u) = G_i$ .*
2. *If  $(E_i \cdot u) \neq \emptyset$ , then  $(E \cdot u) = (E_1 \cdot u)G_1 + \dots + (E_n \cdot u)G_n$ .*
3. *If  $H_i \neq \emptyset$  for all  $i : 1 \leq i \leq n$ , then  $E_1H_1 + \dots + E_nH_n$  is a head/tail form.*
4. *If each  $H_i \neq \emptyset$  and each  $E_i \neq \epsilon$  and for each  $j$  such that  $E_j \neq \emptyset$ ,  $H_j \sim_m G_j$ , then  $E \sim_{m+1} E_1H_1 + \dots + E_nH_n$ .*
5. *If each  $H_i \neq \emptyset$  and for each  $j$  such that  $E_j \neq \emptyset$ ,  $H_j \sim G_j$ , then  $E \sim E_1H_1 + \dots + E_nH_n$ .*

**Proof:** Part 1 is a simple generalisation of proposition 1.1 and is proved in the same way. Part 2 follows from it.  $E_1G_1 + \dots + E_nG_n$  is a head/tail form, and, therefore, by definition each  $G_i \neq \emptyset$  and  $E_1 + \dots + E_n$  is admissible. Therefore, if each  $H_i \neq \emptyset$ , then  $E_1H_1 + \dots + E_nH_n$  is a head/tail form. For 4 assume  $F = E_1H_1 + \dots + E_nH_n$  and  $E \not\sim_{m+1} F$ .  $E$  and  $F$  have the same heads  $E_1 + \dots + E_n$  and each  $E_i \neq \epsilon$ . Therefore, there must be a word  $u$  with  $0 < |u| \leq m + 1$  and  $(E \cdot u) = G_i$  and  $(F \cdot u) = H_i$  and  $G_i \not\sim_{(m+1)-|u|} H_i$  which is a contradiction. The final part uses a similar argument.  $\square$

Two configurations may have the same heads and different tails, or may have the same tails and different heads. If  $E$  has the head/tail form  $E_1G_1 + \dots + E_nG_n$  and  $F$  has a similar head/tail form  $F_1G_1 + \dots + F_nG_n$  involving the same tails<sup>8</sup>, then the imbalance between  $E$  and  $F$ , relative to this presentation, is  $\max \{ |E_i|, |F_i| : 1 \leq i \leq n \}$ . If the imbalance is 0, then they are the same. The next result establishes a key property of a combination of such configurations.

<sup>8</sup>Any pair of configurations  $E$  and  $F$  have a head/tail form involving the same tails:  $E = EG$  and  $F = FG$  when  $G = \epsilon$ .

**Proposition 3** Assume the following configurations.

$$\begin{array}{ll} E & = E_1G_1 + \dots + E_nG_n & F & = F_1G_1 + \dots + F_nG_n \\ E' & = E_1H_1 + \dots + E_nH_n & F' & = F_1H_1 + \dots + F_nH_n \end{array}$$

If  $E \sim_m F$  and  $E' \not\sim_m F'$ , then there is a word  $u$ ,  $|u| \leq m$ , and an  $i$  such that either 1 or 2.

1.  $(E' \cdot u) = H_i$  and  $(F' \cdot u) = (F_1 \cdot u)H_1 + \dots + (F_n \cdot u)H_n$  and  $(F' \cdot u) \neq \emptyset$  and  $(E' \cdot u) \not\sim_{m-|u|} (F' \cdot u)$ ,
2.  $(F' \cdot u) = H_i$  and  $(E' \cdot u) = (E_1 \cdot u)H_1 + \dots + (E_n \cdot u)H_n$  and  $(E' \cdot u) \neq \emptyset$  and  $(E' \cdot u) \not\sim_{m-|u|} (F' \cdot u)$ .

**Proof:** Assume  $E \sim_m F$  and  $E' \not\sim_m F'$ . Therefore, there is a word  $u$ ,  $|u| = m$ , and, without loss of generality,  $(E' \cdot u) = \emptyset$  and  $(F' \cdot u) \neq \emptyset$ , by definition of  $\sim_m$ . However,  $(E \cdot u) = \emptyset$  if, and only if,  $(F \cdot u) = \emptyset$ . Therefore, there must be a smallest prefix  $v$  of  $u$  such that either  $(E' \cdot v) = H_i$  and  $(F' \cdot v) = (F_1 \cdot v)H_1 + \dots + (F_n \cdot v)H_n$ , or  $(F' \cdot v) = H_i$  and  $(E' \cdot v) = (E_1 \cdot v)H_1 + \dots + (E_n \cdot v)H_n$ , and  $(E' \cdot v) \not\sim_{m-|v|} (F' \cdot v)$ . And now the result follows because  $(E \cdot v) \sim_{m-|v|} (F \cdot v)$  and  $E$  has the same head as  $E'$  and  $F$  has the same head as  $F'$  and because  $G_i, H_i \neq \emptyset$  this implies that  $(E' \cdot v), (F' \cdot v) \neq \emptyset$ .  $\square$

### 4.3 The tableau proof system

The procedure for deciding  $E \sim F$  is a tableau proof system as in section 2.3. The proof rules, in figure 14, are generalisations of the unfold and balance rules for simple grammars. UNF reduces a goal  $E \doteq F$  to subgoals  $(E \cdot a) \doteq (F \cdot a)$  for each  $a \in \mathbf{A}$ . Because it is intended that there is a unique tableau associated with any goal, the subgoals are ordered by the ordering on  $\mathbf{A}$ . It is complete and sound (compare section 2.3).

**Fact 1** [Completeness and soundness of UNF]

1. If  $E \sim F$  and  $a \in \mathbf{A}$ , then  $(E \cdot a) \sim (F \cdot a)$ .
2. If  $E \not\sim_{m+1} F$ , then for some  $a \in \mathbf{A}$ ,  $(E \cdot a) \not\sim_m (F \cdot a)$ .

**Example 1** Below is an application of UNF using example 3 of section 3.3.

$$\frac{YX + Z \doteq YYX + YZ + Z}{X \doteq YX + Z \quad \epsilon \doteq \epsilon \quad YYX + YZ + Z \doteq YYYX + YYZ + YZ + Z}$$

The three subgoals are the result after  $a$ ,  $b$  and  $c$ .  $\square$

As in section 2.3, if  $E' \doteq F'$  is a subgoal given by  $m$  consecutive applications of UNF (and no other rule) to  $E \doteq F$ , then there is an *associated* word  $u$  such that  $|u| = m$  and  $E' = (E \cdot u)$  and  $F' = (F \cdot u)$ .

**Fact 2** If  $E' \doteq F'$  is a subgoal that is a result of  $m$  consecutive applications of UNF (and no other rule) to  $E \doteq F$ , then  $|E'| \leq |E| + m$  and  $|F'| \leq |F| + m$ .

**Example 2** An application of BAL(L) uses example 1 of section 2.2.

$$\frac{XXXXXX \doteq AAAAAA}{YXXXXXX \doteq CAAAAA} \text{ UNF} \quad \dots \quad \frac{YXXXXXX \doteq CAAAAA}{YXAAAAA \doteq CAAAAA} \text{ BAL(L)}$$

UNF

$$\frac{E \doteq F}{(E \cdot a_1) \doteq (F \cdot a_1) \quad \dots \quad (E \cdot a_k) \doteq (F \cdot a_k)} \mathbf{A} = \{a_1, \dots, a_k\}$$

BAL(R)

$$\frac{\begin{array}{c} F \doteq X_1 H_1 + \dots + X_k H_k \\ \vdots \\ F' \doteq E_1 H_1 + \dots + E_k H_k \end{array}}{F' \doteq E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k))} \mathbf{C}$$

BAL(L)

$$\frac{\begin{array}{c} X_1 H_1 + \dots + X_k H_k \doteq F \\ \vdots \\ E_1 H_1 + \dots + E_k H_k \doteq F' \end{array}}{E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k)) \doteq F'} \mathbf{C}$$

where C is the condition

1. Each  $E_i \neq \varepsilon$  and at least one  $H_i \neq \varepsilon$ .
2. There are precisely  $\max\{|w(X_i)| : E_i \neq \emptyset \text{ for } 1 \leq i \leq k\}$  applications of UNF between the top goal and the bottom goal, and no application of any other rule.
3. If  $u$  is the word associated with the sequence of UNFs, then  $E_i = (X_i \cdot u)$  for each  $i : 1 \leq i \leq k$ .

Figure 14: The tableau proof rules

The second goal is the result of UNF when the label is  $a$  (and the other subgoal for  $b$  is omitted).  $w(X) = b$ , so  $m = 1$ . Therefore, BAL(L) applies to the second goal:  $X_1 = X$ ,  $H_1 = XXXXX$ ,  $E_1 = YX$  and  $F = AAAAAA$ . So  $H_1$  is replaced with  $(F \cdot b) = AAAAAA$ . The imbalance between configurations of the last goal is 2. The same bound on imbalance occurs for a starting goal  $X^n \doteq A^n$  when  $n > 0$ .  $\square$

**Definition 1** An application of BAL *use*  $F$  if  $F$  is the configuration in the initial goal of the rule, as in figure 14.

Proposition 1.2, below, captures precisely the bounds of imbalance: the statement is for BAL(L), and its symmetric version covers BAL(R).

**Proposition 1** *Assume  $E' \doteq F'$  is the result of BAL(L) using  $F$ .*

1.  $|E'| \leq |F| + 2M + 1$  and  $|F'| \leq |F| + M$ .
2. *If  $F = \beta_1 G_1 + \dots + \beta_n G_n$  is in  $m$ -head form and  $m \geq M$ , then  $E' = E_1 G_1 + \dots + E_n G_n$  and each  $|E_i| \leq m + 2M + 1$  and  $F' = F_1 G_1 + \dots + F_n G_n$  and  $|F_i| \leq m + M$ .*

**Proof:** Part 1 is straightforward, and uses fact 2, above, and proposition 1.1 of the previous section. For part 2 assume an application of BAL(L) with initial goal  $X_1 H_1 + \dots + X_k H_k \doteq F$ . Let  $F$  be presented in  $m$ -head form,  $\beta_1 G_1 + \dots + \beta_n G_n$ , where  $m \geq M$ . Assume that  $u$ ,  $|u| \leq M$ , is the word associated with the sequence of applications of UNF before BAL(L) is applied, so  $|u| = \max\{|w(X_i)| : (X_i \cdot u) \neq \emptyset\}$ . The result of BAL(L),  $E' \doteq F'$ , is the following goal.

$$(X_1 \cdot u)(F \cdot w(X_1)) + \dots + (X_k \cdot u)(F \cdot w(X_k)) \doteq (\beta_1 \cdot u)G_1 + \dots + (\beta_n \cdot u)G_n$$

Because  $|w(X_i)| \leq m$  and  $F$  is in  $m$ -head form, the left configuration has the matrix form

$$\begin{aligned} (X_1 \cdot u)(F \cdot w(X_1)) &= (X_1 \cdot u)(\beta_1 \cdot w(X_1))G_1 + \dots + (X_1 \cdot u)(\beta_n \cdot w(X_1))G_n \\ &\vdots \\ (X_k \cdot u)(F \cdot w(X_k)) &= (X_k \cdot u)(\beta_1 \cdot w(X_k))G_1 + \dots + (X_k \cdot u)(\beta_n \cdot w(X_k))G_n. \end{aligned}$$

The heads  $X_1 + \dots + X_k$  and  $\beta_1 + \dots + \beta_n$  are admissible, so  $(X_1 \cdot u) + \dots + (X_k \cdot u)$  and  $(\beta_1 \cdot w(X_i)) + \dots + (\beta_n \cdot w(X_i))$  are also admissible. Let  $E_i$  be the sum of the heads of the  $i$ th column, without the tail  $G_i$ ,

$$(X_1 \cdot u)(\beta_i \cdot w(X_1)) + \dots + (X_k \cdot u)(\beta_i \cdot w(X_k)).$$

Therefore each  $E_i$  is admissible and  $E_1 + \dots + E_n$  is admissible. Consequently,  $E_1 G_1 + \dots + E_n G_n$  is a valid head/tail form. Let  $F_i = (\beta_i \cdot u)$  for each  $i : 1 \leq i \leq n$ . Therefore, the result of BAL(L) is  $E_1 G_1 + \dots + E_n G_n \doteq F_1 G_1 + \dots + F_n G_n$  that has bounded imbalance, using proposition 1.1, of the previous section,  $|E_i| \leq m + 2M + 1$  and  $|F_i| \leq m + M$ .  $\square$

The BAL rules are sound and complete (compare section 2.3).

**Proposition 2** [Completeness and soundness of BAL]

1. *If  $X_1 H_1 + \dots + X_k H_k \sim F$  and  $E_1 H_1 + \dots + E_k H_k \sim F'$ , then  $E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k)) \sim F'$ .*
2. *If  $X_1 H_1 + \dots + X_k H_k \sim_{n+m} F$  and  $E_1 H_1 + \dots + E_k H_k \not\sim_{n+1} F'$  and each  $E_i \neq \varepsilon$  and  $m \geq \max\{|w(X_i)| : E_i \neq \emptyset\}$ , then  $E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k)) \not\sim_{n+1} F'$ .*

$$\begin{array}{rcl}
& & F \\
& & \vdots \\
E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k)) & \doteq & H \quad \text{BAL(L)} \\
& & \vdots \\
(F \cdot w(X_i)) = G_1 & \doteq & H_1 \quad \text{UNFs} \\
& & \vdots \\
G_k & \doteq & H_k
\end{array}$$

Figure 15: A potential switch from BAL(L) to BAL(R)

**Proof:** Let  $E$  be  $X_1H_1 + \dots + X_kH_k$  and assume  $E \sim F$ . By proposition 1.1 of the previous section,  $(E \cdot w(X_i)) = H_i$ , so  $H_i \sim (F \cdot w(X_i))$ . By assumption of head/tail form, each  $H_i \neq \emptyset$  and, therefore, each  $(F \cdot w(X_i)) \neq \emptyset$ . Consequently,  $E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k)) \sim E_1H_1 + \dots + E_kH_k$  and the result thereby follows. For part 2, assume  $X_1H_1 + \dots + X_kH_k \sim_{n+m} F$  and  $E_1H_1 + \dots + E_kH_k \not\sim_{n+1} F'$ , and each  $E_i \neq \varepsilon$ . If  $E_i \neq \emptyset$ , then  $|E_i| > 0$ . Because  $(X_1H_1 + \dots + X_kH_k) \cdot w(X_i) = H_i$  and  $|w(X_i)| \leq m$ ,  $(F \cdot w(X_i)) \sim_n H_i$ , so  $E_1(F \cdot w(X_1)) + \dots + E_n(F \cdot w(X_n)) \sim_{n+1} E_1H_1 + \dots + E_nH_n$ , and the result follows.  $\square$

There is no rule corresponding to CUT. Indeed, it is not clear how this rule for simple grammars could be generalised to admissible configurations.

It is intended that there be a unique tableau associated with any initial goal. So restrictions will be placed on which rule is to be applied when. First, there is an issue as to the initial premise of an application of BAL. It is possible that BAL applies to a goal, but with respect to more than one initial premise; see example 3 below. In any choice of initial premise, there can be only one premise that does not occur above the others in the proof tree: it is this premise that will be assumed. That is, the initial premise of a BAL is the one that is “closest” to the goal and, therefore, the one that involves the least number of applications of UNF. To resolve which rule should be applied, the following priority order is assumed.

1. If BAL(L) is permitted, then apply BAL(L)
2. If BAL(R) is permitted, then apply BAL(R)
3. Otherwise, apply UNF

However, whether an application of BAL is permitted involves more than fulfillment of the side condition. It also depends on the previous application of a BAL.

Initially, either BAL is permitted provided that its side condition is true. If an application of BAL uses  $F$ , then the resulting goal contains the configuration  $E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k))$ .  $E_i$  is a “top” of the application of BAL and  $(F \cdot w(X_i))$  is a “bottom”. Assume an application of BAL(L). A subsequent application of BAL(L) is permitted provided the side condition of the rule is fulfilled. However, BAL(R) is not permitted until a bottom of the previous application of BAL(L) is exposed and the side condition of the rule is true. Between the application of BAL(L) that uses  $F$  and the goal  $G_1 \doteq H_1$  in figure 15, there are no other applications of BAL(L), and  $G_1$  is a bottom,  $(F \cdot w(X_i))$ , of the application of BAL(L). BAL(R) is now permitted provided it uses configuration  $G_i$ ,  $i \geq 1$ , and the side condition holds. BAL(R) is not permitted using a configuration from a goal above  $G_1 \doteq H_1$ , even when the side condition is true. The strategy is to apply

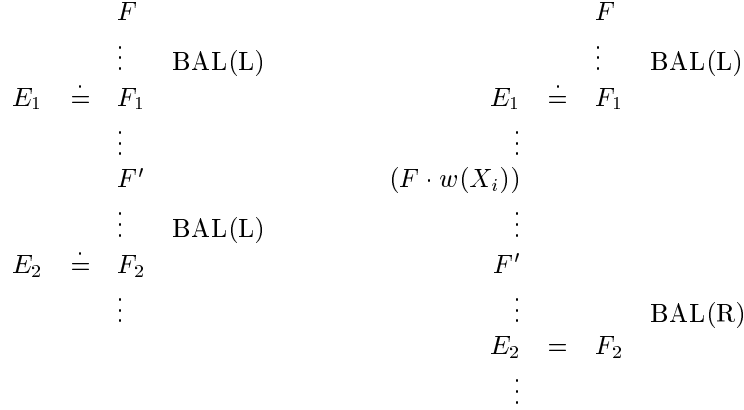


Figure 16: Proof of Proposition 3

a BAL rule whenever it is permitted, and if both BAL rules are permitted, then priority lies with BAL(L). If BAL(R) is applied, then the strategy is to repeatedly apply BAL(R), and to use UNF otherwise. BAL(L) is only permitted once a bottom of the previous application of BAL(R) becomes the right hand configuration of a goal and the side condition holds.

For the purpose of exposition, the tableaux proof rules have been presented in two stages: first, as rules and then with a priority order. However, the priority order could be formalised as side conditions of the rules<sup>9</sup>. The consequence is that when building a tableau proof tree, there is just one choice of which rule to apply next to any subgoal.

A branch of a tableau from a subgoal  $g(0)$  is a sequence of goals that start from  $g(0)$ . The following result motivates the restriction on the application of a BAL rule.

**Proposition 3** *If there are consecutive applications of BAL in a branch that use  $F$  and  $F'$ , then there is a word  $u$  such that  $F' = (F \cdot u)$ .*

**Proof:** Assume consecutive applications of BAL that use  $F$  and  $F'$ , and without loss of generality assume that the application using  $F$  is BAL(L). There are two possibilities, that the application using  $F'$  is again BAL(L), shown on the left in Figure 16, and that it is BAL(R), shown on the right in figure 16. In both cases,  $E_1 \doteq F_1$  is the result of BAL using  $F$  and  $E_2 \doteq F_2$  is the result of BAL using  $F'$ . Consider the first case. There are only applications of UNF between  $F$  and the goal  $E' \doteq F_1$  immediately before  $E_1 \doteq F_1$ . Therefore, there is a word  $u_1$  such that  $(F \cdot u_1) = F_1$ . Between  $E_1 \doteq F_1$  and the goal containing  $F'$  there are only applications of UNF, and, therefore, there is a word  $u_2$  such that  $(F_1 \cdot u_2) = F'$ . Therefore,  $F' = (F \cdot u_1 u_2)$ . For the second case, there are only applications of UNF between the goal with left configuration  $(F \cdot w(X_i))$  and the goal with left configuration  $F'$ . Therefore, there is a word  $u_2$  such that  $(F \cdot w(X_i) u_2) = F'$ .  $\square$

**Corollary 1** *If  $F_0, F_1, \dots, F_n$  are successive configurations used in applications of BAL in a branch, then there are words  $u_1, \dots, u_n$  such that  $F_i = (F_0 \cdot u_1 \dots u_i)$ .*

**Example 3** An initial part of the tableau, continuing on from example 2 above,

<sup>9</sup>For instance, the additional side condition of a BAL(R) is that BAL(L) does not apply and either there is no previous application of a BAL, or the previous application of a BAL is a BAL(R), or the previous application of a BAL is a BAL(L) and a bottom configuration occurs at, or above, the initial premise.





### Successful final goals

$$\begin{array}{rcl}
 & E_1 \doteq F_1 & \text{the root goal} \\
 & \vdots & \\
 & E_n \doteq F_n & \text{obeys extension theorem} \\
 E \doteq E & & 
 \end{array}$$

Figure 18: Successful final goals

The extension is (A):  $g(2)$  extends  $g(1)$  by (A) and  $h(2)$  extends  $h(1)$  by (A). The theorem provides the following result: for any  $m$ , if  $YXA^5 \sim_m CA^5$  and  $YXA^6 \sim_m CA^6$ , then  $YXA^7 \sim_m CA^7$  which justifies that  $YXA^7 \doteq CA^7$  is a successful final goal. The argument is similar to theorem 2 of section 2.5 for a repeat goal<sup>10</sup>.

**Definition 2** Assume a branch of goals  $d(0), \dots, d(l)$ . The goal  $d(l)$ ,  $E_1H_1 + \dots + E_nH_n \doteq F_1H_1 + \dots + F_nH_n$ , obeys the extension theorem if the following hold

1. There are families of goals  $g(i)$ ,  $h(i)$ ,  $1 \leq i \leq 2^n$  belonging to  $\{d(0), \dots, d(l)\}$ , and each goal  $g(i)$  has the form  $E_1G_1^i + \dots + E_nG_n^i \doteq F_1G_1^i + \dots + F_nG_n^i$  and each goal  $h(i)$  has the form  $E_1H_1^i + \dots + E_nH_n^i \doteq F_1H_1^i + \dots + F_nH_n^i$ .
2. The goal  $h(2^n)$  is  $d(l)$  and there is at least one application of UNF between goal  $h(2^n - 1)$  and  $d(l)$ .
3. There are extensions  $e_1, \dots, e_n$  such that for each  $e_j$  and  $i \geq 0$

$$\begin{array}{l}
 g(2^j i + 2^{j-1} + 1) \text{ extends } g(2^j i + 2^{j-1}) \text{ by } e_j \\
 h(2^j i + 2^{j-1} + 1) \text{ extends } h(2^j i + 2^{j-1}) \text{ by } e_j.
 \end{array}$$

The second occurrence of a repeat goal in a branch obeys the extension theorem provided that there is at least one application of UNF between the two occurrences. Assume it has the form  $E_1G_1^i + \dots + E_nG_n^i \doteq F_1G_1^i + \dots + F_nG_n^i$ . Except for  $h(2^n)$ , the goals  $g(i)$  and  $h(i)$  are its first occurrence and each extension is the identity, ( $\epsilon$ ). All three conditions of definition 2 are thereby satisfied.

The definition of successful final goal is given in figure 18.

The deterministic procedure that decides whether  $E \sim F$  is straightforward, and is defined iteratively.

1. Stage 0: start with the root goal  $g(0)$ ,  $E \doteq F$ , that becomes a frontier node of the branch  $g(0)$ .
2. Stage  $n + 1$ : if a current frontier node  $g(n)$  of branch  $g(0), \dots, g(n)$  is an unsuccessful final goal, then halt and return “unsuccessful tableau”; if each frontier node  $g(n)$  of branch  $g(0), \dots, g(n)$  is a successful final goal, then return “successful tableau”; otherwise, for each frontier node  $g(n)$  of branch  $g(0), \dots, g(n)$  that is not a final goal, apply the next rule to it, and the subgoals that result are the new frontier nodes of the extended branches.

<sup>10</sup>Assume that the branch involving the goals  $g(1)$ ,  $g(2)$ , and  $h(2)$  is an offending branch, then there are  $m_1$ ,  $m_2$  and  $m_3$  with  $m_1 > m_2 > m_3$  such that  $g(1)$  is true at level  $m_1$  and false at level  $m_1 + 1$ ,  $g(2)$  is true at level  $m_2$  and false at level  $m_2 + 1$  and  $h(2)$  is true at  $m_3$  and false at  $m_3 + 1$ . But, this leads to a contradiction, let  $m = m_3 + 1$ , both  $g(1)$  and  $g(2)$  are true at level  $m$  and, therefore, by the extension theorem, so is  $h(2)$ .

The following show decidability of the DPDA equivalence problem and establish a complexity upper bound that is primitive recursive. The proofs are presented later.

**Theorem 2** [Soundness and completeness of decision procedure]

1. If  $E \not\sim F$ , then the decision procedure terminates with “unsuccessful tableau”.
2. If  $E \sim F$ , then the decision procedure terminates with “successful tableau”.

A DPDA with  $s_1$  stack symbols and  $p$  states is transformed into an SDG with at most  $s = s_1 \times p^2$  stack symbols and whose largest partition is at most  $p$ . Moreover, a configuration  $p\alpha$  of the DPDA where  $|\alpha| \leq n$  is transformed into an SDG configuration  $\text{sum}(p\alpha)$  where  $|\text{sum}(p\alpha)| \leq n$ . For the complexity upper bound, some basis functions are introduced assuming a fixed SDG with  $s$  stack symbols and largest partition  $p$ . Let  $\text{goal}(h)$  be the number of different goals  $E \doteq F$  such that  $|E|, |F| \leq h$ . The function  $\text{width}(h)$  is the maximum  $n$  of an admissible configuration  $\beta_1 + \dots + \beta_n$  such that  $|\beta_i| \leq h$  (so,  $\text{width}(h) \leq p^h$ ). The function  $\text{ext}(d, w)$  is the number of different extensions  $e$  such that  $|e| \leq d$  and  $e$  has width at most  $w$  (so,  $\text{ext}(d, w) \leq 2^{s \cdot d \cdot w}$ ). The other measure used is the maximum norm  $M$  that is bounded by  $2^s$ . Some auxiliary functions are now defined from the basis functions.

**Definition 3**

1. The function  $f_1(k) = M(k + 1 + M^2 + 2M) + \text{goal}(M)$ .
2. The function  $f_2[d, h, w](n)$  is defined recursively.

$$\begin{aligned} f_2[d, h, w](0) &= \text{goal}(h) \\ f_2[d, h, w](j + 1) &= \text{ext}(f_2[d, h, w](j) \times d, w) \times f_2[d, h, w](j) \end{aligned}$$

3. Let  $d = M^2 + 2M$ ,  $h = M^2 + 5M + 2$ ,  $w = \text{width}(h)$  and  $b = f_2[d, h, w](w)$ . Then,  $f(n) = n \times d^b \times f_1(n + bd)$ .

**Theorem 3** If  $|E|, |F| < n$ , then the decision procedure with root  $E \doteq F$  terminates within  $f(n)$  stages.

**Corollary 1** If  $|E|, |F| < n$ , then  $E \sim F$  if, and only if,  $E \sim_{f(n)} F$ .

Alternatively, this result can be directly stated for DPDA.

**Corollary 2** If  $|\alpha|, |\beta| < n$ , then  $p\alpha \sim q\beta$  if, and only if,  $p\alpha \sim_{f(n)} q\beta$ .

The bound  $f(n)$  is elementary with respect to  $n$ , the size of the starting goal, but it is only primitive recursive with respect to the size of the DPDA (or, SDG) and, in particular, with respect to the number of states of the DPDA. Much more work is needed to check if this bound is anywhere near optimal. The explosiveness of  $f$  has some intuitive basis when considering application of the extension theorem to a branch. If the stack size of a DPDA increases, the width,  $\text{width}(h)$ , remains the same and so does the maximum number of different goals needed to apply the extension theorem. If the state size of a DPDA increases by 1, then so does the width and the maximum number of different goals needed for an application of the extension theorem quadruples.

## 4.5 Exercises

**Exercise 1** Consider the following DPDA from Sénizergues [26].

$$\begin{array}{lll}
 pX \xrightarrow{a} p_1\epsilon & pX \xrightarrow{b} p_2\epsilon & pX \xrightarrow{c} pXX \\
 p_1X \xrightarrow{a} p_1\epsilon & p_2X \xrightarrow{\epsilon} p_2\epsilon & \\
 qY \xrightarrow{a} q_1YY & qY \xrightarrow{b} q_2\epsilon & qY \xrightarrow{c} qYY \\
 q_1Y \xrightarrow{a} q_1\epsilon & q_2Y \xrightarrow{\epsilon} q_2\epsilon & 
 \end{array}$$

The transition graphs  $G(pX)$  and  $G(qY)$  are pictured below (and their collapsed graphs are easy to imagine).

$$\begin{array}{ccccccc}
 p_1\epsilon & \xleftarrow{a} & p_1X & \xleftarrow{a} & p_1XX & \xleftarrow{a} & \dots \\
 \uparrow a & & \uparrow a & & \uparrow a & & \\
 pX & \xrightarrow{c} & pXX & \xrightarrow{c} & pXXX & \xrightarrow{c} & \dots \\
 \downarrow b & & \downarrow b & & \downarrow b & & \\
 p_2\epsilon & \xleftarrow{\epsilon} & p_2X & \xleftarrow{\epsilon} & p_2XX & \xleftarrow{\epsilon} & \dots \\
 \\
 q_1\epsilon & \xleftarrow{a} & q_1Y & \xleftarrow{a} & q_1YY & \xleftarrow{a} & q_1YYY & \xleftarrow{a} & \dots \\
 & & \uparrow a & & \uparrow a & & \uparrow a & & \\
 & & qY & \xrightarrow{c} & qYY & \xrightarrow{c} & qYYY & \xrightarrow{c} & \dots \\
 & & \downarrow b & & \downarrow b & & \downarrow b & & \\
 & & q_2\epsilon & \xleftarrow{\epsilon} & q_2Y & \xleftarrow{\epsilon} & q_2YY & \xleftarrow{\epsilon} & \dots
 \end{array}$$

$L(pX) = \{c^n b : n \geq 0\} \cup \{c^n a^{n+1} : n \geq 0\}$  and  $L(qY) = \{c^n b : n \geq 0\} \cup \{c^n a^{n+3}\}$ . These configurations can be “equalised” by adding a new stack symbol  $Z$ .

$$\begin{array}{lll}
 p_1Z \xrightarrow{a} p_3Z & p_3Z \xrightarrow{a} p_4Z & p_4Z \xrightarrow{a} p_4\epsilon \\
 p_2Z \xrightarrow{\epsilon} p_2\epsilon & q_1Z \xrightarrow{a} q_1\epsilon & q_2Z \xrightarrow{\epsilon} q_2\epsilon
 \end{array}$$

It now follows that  $pXZ \sim qYZ$ .

- Translate the full DPDA into an SDG.
- Give the successful tableau for  $pXZ \sim qYZ$ . □

more to be added

## 5 Proofs

### 5.1 Proof of the extension theorem

The extension theorem, theorem 1 of section 4.4, is a corollary of a more general result.

**Definition 1** Assume a family of goals  $E_1G_1^s + \dots + E_nG_n^s \doteq F_1G_1^s + \dots + F_nG_n^s$  with the same heads where  $s \in I$ .

1. A “known” at level  $m$  is either
  - (a)  $G_i^s \sim_m G_j^s$  for some  $j < i$  and for all  $s \in I$ , or
  - (b)  $G_i^s \sim_m J_1G_1^s + \dots + J_nG_n^s$  for all  $s \in I$  and each  $J_k \neq \varepsilon$ .
2. Two knowns  $G_i^s \sim_m H_1^s$  and  $G_j^s \sim_m H_2^s$  are distinct, if  $i \neq j$ .

**Theorem 1** [The generalised extension theorem] *Assume there are two families of goals  $g(i), h(i), 1 \leq i \leq 2^{n-k}$ , and each goal  $g(i)$  has the form  $E_1G_1^i + \dots + E_nG_n^i \doteq F_1G_1^i + \dots + F_nG_n^i$  and each goal  $h(i)$  has the form  $E_1H_1^i + \dots + E_nH_n^i \doteq F_1H_1^i + \dots + F_nH_n^i$ , and  $k$  is the number of distinct knowns for the family  $g(i) \cup h(i)$  at level  $m$ . Assume extensions  $e_1, \dots, e_{n-k}$  such that for each  $e_j$  and  $i \geq 0$*

$$\begin{aligned} g(2^j i + 2^{j-1} + 1) &\text{ extends } g(2^j i + 2^{j-1}) \text{ by } e_j \\ h(2^j i + 2^{j-1} + 1) &\text{ extends } h(2^j i + 2^{j-1}) \text{ by } e_j. \end{aligned}$$

*If each goal  $g(i)$  is true at level  $m, i : 1 \leq i \leq 2^{n-k}$ , and each goal  $h(j), j : 1 \leq j < 2^{n-k}$ , is true at level  $m$ , then  $h(2^{n-k})$  is true at level  $m$ .*

**Proof:** The proof is by induction on  $n - k$ . For the base case assume  $n - k = 0$ . Assume that there are two goals  $g(1), E_1G_1 + \dots + E_nG_n \doteq F_1G_1 + \dots + F_nG_n$ , abbreviated to  $E \doteq F$ , and  $h(1), E_1H_1 + \dots + E_nH_n \doteq F_1H_1 + \dots + F_nH_n$ , abbreviated to  $E' \doteq F'$ , and  $n$  distinct knowns for  $\{g(1), h(1)\}$  at level  $m$ . That is, for each  $i : 1 \leq i \leq n$  either  $G_i \sim_m G_j$  and  $H_i \sim_m H_j$  and  $j < i$ , or  $G_i \sim_m J_1G_1 + \dots + J_nG_n$  and  $H_i \sim_m J_1H_1 + \dots + J_nH_n$  where  $J_i \neq \varepsilon$ . We prove that if  $g(1)$  is true at level  $m, E \sim_m F$ , then  $h(1)$  is also true at level  $m, E' \sim_m F'$ . Suppose not. Then, without loss of generality, by proposition 3 of section 4.2 there is a word  $u, |u| \leq m$ , and  $(E' \cdot u) = H_i$  and  $(F' \cdot u) = (F_1 \cdot u)H_1 + \dots + (F_n \cdot u)H_n$  and  $(F' \cdot u) \neq \emptyset$  and  $(E' \cdot u) \not\sim_{m-|u|} (F' \cdot u)$ . However, because  $E \sim_m F$  it follows that  $G_i \sim_{m-|u|} (F_1 \cdot u)G_1 + \dots + (F_n \cdot u)G_n$ . The first case is that  $G_i \sim_m G_j$  and  $H_i \sim_m H_j$  where  $j < i$ . Therefore, using properties of  $\sim_k, G_j \sim_{m-|u|} (F_1 \cdot u)G_1 + \dots + (F_n \cdot u)G_n$  and  $H_j \not\sim_{m-|u|} (F_1 \cdot u)H_1 + \dots + (F_n \cdot u)H_n$ . This can only be repeated finitely many times,  $G_j \sim_m G_{j'}$  and  $H_j \sim_m H_{j'}$ , and  $j' < j$ . Hence, the second case must then occur,  $G_i \sim_m J_1G_1 + \dots + J_nG_n$  and  $H_i \sim_m J_1H_1 + \dots + J_nH_n$  and no  $J_i = \varepsilon$ .  $J_1G_1 + \dots + J_nG_n \sim_{m-|u|} (F_1 \cdot u)G_1 + \dots + (F_n \cdot u)G_n$  and  $J_1H_1 + \dots + J_nH_n \not\sim_{m-|u|} (F_1 \cdot u)H_1 + \dots + (F_n \cdot u)H_n$ . Now proposition 3 of section 4.2 is applied again. Therefore, by repeating this argument a contradiction will be obtained.

For the general step, assume that it holds for  $n - k < t$  and consider  $n - k = t$ . So, there are two families of goals  $g(i), h(i), 1 \leq i \leq 2^{n-k}$ , and each goal  $g(i)$  has the form  $E_1G_1^i + \dots + E_nG_n^i \doteq F_1G_1^i + \dots + F_nG_n^i$  and each goal  $h(i)$  has the form  $E_1H_1^i + \dots + E_nH_n^i \doteq F_1H_1^i + \dots + F_nH_n^i$ , and  $k$  is the number of distinct knowns for the family  $g(i) \cup h(i)$  at level  $m$ . Assume extensions  $e_1, \dots, e_{n-k}$  such that for each  $e_j$  and  $i \geq 0$

$$\begin{aligned} g(2^j i + 2^{j-1} + 1) &\text{ extends } g(2^j i + 2^{j-1}) \text{ by } e_j \\ h(2^j i + 2^{j-1} + 1) &\text{ extends } h(2^j i + 2^{j-1}) \text{ by } e_j. \end{aligned}$$

Assume that each goal  $g(i)$  is true at level  $m$ ,  $i : 1 \leq i \leq 2^{n-k}$ , and each goal  $h(j)$ ,  $j : 1 \leq j < 2^{n-k}$ , is true at level  $m$ . The aim is to show that  $h(2^{n-k})$  is also true at level  $m$ . Suppose not. Let  $q = 2^{n-k}$ . Therefore,  $E_1 H_1^q + \dots + E_n H_n^q \not\sim_m F_1 H_1^q + \dots + F_n H_n^q$  and  $E_1 G_1^q + \dots + E_n G_n^q \sim_m F_1 G_1^q + \dots + F_n G_n^q$ . Without loss of generality, by proposition 3 of section 4.2 there is a word  $u$ ,  $|u| \leq m$ , and  $(E_1 H_1^q + \dots + E_n H_n^q \cdot u) = H_i^q$  and  $(F_1 H_1^q + \dots + F_n H_n^q \cdot u) = (F_1 \cdot u) H_1^q + \dots + (F_n \cdot u) H_n^q$  and  $H_i^q \not\sim_{m-|u|} (F_1 \cdot u) H_1^q + \dots + (F_n \cdot u) H_n^q$  and for all  $j : 1 \leq j \leq q$ ,  $G_i^j \sim_{m-|u|} (F_1 \cdot u) G_1^j + \dots + (F_n \cdot u) G_n^j$  and for all  $j : 1 \leq j < q$ ,  $H_i^j \sim_{m-|u|} (F_1 \cdot u) H_1^j + \dots + (F_n \cdot u) H_n^j$ . There are two cases. First is that each  $G_i^j$  and  $H_i^j$ ,  $1 \leq j \leq q$ , is a known at level  $m$ . The proof proceeds as in the base case, but here with respect to the whole family of goals. Therefore, at some point the second case happens, that  $G_i^j$  and  $H_i^j$  are not knowns. However, each even goal  $g(2j)$  and  $h(2j)$  extends the previous goal  $g(2j-1)$  and  $h(2j-1)$  by  $e_1$ . Therefore, each  $G_i^{2j} \sim_{m-|u|} (F_1 \cdot u) G_1^{2j} + \dots + (F_n \cdot u) G_n^{2j}$  becomes  $J_1 G_1^{2j-1} + \dots + J_n G_n^{2j-1} \sim_{m-|u|} F'_1 G_1^{2j-1} + \dots + F'_n G_n^{2j-1}$  by substituting in the entries of  $e_1$ , and  $J_1 H_1^{2j-1} + \dots + J_n H_n^{2j-1} \sim_{m-|u|} F'_1 H_1^{2j-1} + \dots + F'_n H_n^{2j-1}$  when  $j < (q-1)$  and  $J_1 H_1^{q-1} + \dots + J_n H_n^{q-1} \not\sim_{m-|u|} F'_1 H_1^{q-1} + \dots + F'_n H_n^{q-1}$ . Let these goals be  $g'(i)$ ,  $h'(i)$ ,  $1 \leq i \leq 2^{n-(k+1)}$  and consider the extensions  $e'_1, \dots, e'_{n-(k+1)}$  where  $e'_i = e_1 e_{i+1}$ . It follows that for each  $e'_j$  and  $i \geq 0$ ,  $g'(2^j i + 2^{j-1} + 1)$  extends  $g'(2^j i + 2^{j-1})$  by  $e'_j$  and the same for the  $h'(i)$ s. Moreover, the number of knowns is now  $(k+1)$  because the previous  $k$  knowns at level  $m$  remain knowns at level  $m - |u|$  and there is the new known at level  $m - |u|$  because for each  $j$ ,  $G_i^j \sim_{m-|u|} (F_1 \cdot u) G_1^j + \dots + (F_n \cdot u) G_n^j$  and  $H_i^j \sim_{m-|u|} (F_1 \cdot u) H_1^j + \dots + (F_n \cdot u) H_n^j$ . Therefore, by the induction hypothesis if every goal  $g'(i)$ ,  $1 \leq i \leq 2^{n-(k+1)}$  is true at level  $m - |u|$  and every goal  $h'(i)$ ,  $1 \leq i < 2^{n-(k+1)}$ , is true at level  $m - |u|$ , then  $h'(2^{n-(k+1)})$  is also true at level  $m - |u|$ , which contradicts that  $J_1 H_1^{q-1} + \dots + J_n H_n^{q-1} \not\sim_{m-|u|} F'_1 H_1^{q-1} + \dots + F'_n H_n^{q-1}$ .  $\square$

## 5.2 Correctness of the decision procedure

In this section theorems 2 and 3 of section 4.4 are shown. Part 1 of theorem 2 is straightforward.

**Theorem 1** *If  $E \not\sim F$ , then the decision procedure terminates with “unsuccessful tableau”.*

**Proof:** Assume  $E \not\sim F$ . If one of  $E, F$  is  $\emptyset$  and the other isn't, then the decision procedure terminates at stage 0 with “unsuccessful tableau”. Otherwise, both  $E$  and  $F$  are not  $\emptyset$ , and there is a least  $n$  such that  $E \not\sim_n F$ . The proof rules are sound. Therefore, at each stage of the decision procedure there is at least one offending branch of false goals such that the falsity index decreases by one each time UNF is applied. At some stage,  $m \geq n - 1$ , there is a frontier goal  $E_m \not\sim_1 F_m$  and UNF is applied to it. For some  $a$  either  $(E_m \cdot a) = \emptyset$  and  $(F_m \cdot a) \neq \emptyset$ , or  $(F_m \cdot a) = \emptyset$  and  $(E_m \cdot a) \neq \emptyset$ , so there is an unsuccessful final goal at stage  $m + 1$ , and the decision procedure returns “unsuccessful tableau”.  $\square$

Part 2 of theorem 2 is more intricate. Assume that  $E \sim F$ . The decision procedure needs to output “successful tableau”. The tableau proof rules preserve truth, so at each stage of the decision procedure every frontier goal is true. Therefore, it is not possible to have an unsuccessful final goal. It is also not possible to become stuck because UNF can always apply. The only issue is that the construction goes on forever. If it does, then there is an infinite branch of goals,  $g(0), \dots, g(n), \dots$  where  $g(0)$  is the root goal  $E \doteq F$ . A careful analysis of sequences of goals in a branch is now presented that shows that this is impossible. In the following result,

symmetric properties hold if BAL(R) replaces BAL(L) in parts 2 and 4.

**Proposition 1** *Assume that  $g(0), \dots, g(n)$  is a sequence of goals in a branch of a tableau and  $g(0)$  is  $E \dot{=} F$ .*

1. *If  $g(0)$  is the root goal and  $E = \beta_1 G_1 + \dots + \beta_m G_m$  and  $F = \delta_1 H_1 + \dots + \delta_l H_l$  are in  $k$ -head form and  $|E|, |F| > k$ , and there is no application of BAL between  $g(0)$  and  $g(kM)$ , then there are goals  $g(i)$  and  $g(j)$ ,  $1 \leq i, j \leq kM$ , such that  $g(i)$  is  $G_{i'} \dot{=} F'$  and  $g(j)$  is  $E' \dot{=} H_{j'}$  for some  $i'$  and  $j'$ .*
2. *If  $g(0)$  is a result of BAL(L) using  $F'$  and there is no application of a BAL between  $g(0)$  and  $g(k)$  where  $k = M^2 + M$  and  $E = E_1(F' \cdot w(X_1)) + \dots + E_k(F' \cdot w(X_k))$ , then there is a goal  $g(i)$ ,  $i : 1 \leq i \leq k$ , that has the form  $(F' \cdot w(X_j)) \dot{=} F''$ .*
3. *If  $g(0)$  is a result of BAL using  $F'$  and the next application of BAL is  $g(k)$  using  $F''$ , then  $|F''| \leq |F'| + M^2 + 2M$ .*
4. *If  $g(0)$  is a result of BAL(L) using  $F'$  and the next application of BAL is  $g(k)$  using  $F''$  and  $k \geq M^3 + 3M^2 + 3M$ , then  $|F''| < |F'|$ .*

**Proof:** For 1, assume  $g(0)$  is the root goal  $E \dot{=} F$  and there is no application of BAL between  $g(0)$  and  $g(kM)$ . Both BAL rules are permitted provided their side conditions are fulfilled. Let  $E = X_1 G_1^1 + \dots + X_m G_m^1$  and  $F = Y_1 H_1^1 + \dots + Y_l H_l^1$  be in 1-head form. Within  $M$  steps there must be a goal  $E_i \dot{=} G_{i'}^1$  for some  $i'$ , otherwise BAL(L) can apply, and a goal  $F_j \dot{=} H_{j'}^1$  for some  $j'$ , otherwise BAL(R) can apply. The argument is now repeated for these goals on  $G_{i'}^1$  and  $H_{j'}^1$  in 1-head form and a further  $M$  steps, and so on upto  $k$ . For 2, assume  $g(0)$ ,  $E \dot{=} F$ , is the result of BAL(L) using  $F'$  and  $E = E_1(F' \cdot w(X_1)) + \dots + E_k(F' \cdot w(X_k))$  and there is no application of a BAL between  $g(0)$  and  $g(k)$  where  $k = M^2 + M$ . BAL(L) is permitted throughout this branch provided its side condition holds. Each  $|E_i| \leq M + 1$  because there are at most  $M$  applications of UNF between the goal with  $F'$  and  $g(0)$ . Hence, using the argument from part 1, within  $M^2 + M$  steps there is a goal  $g(i)$  whose left configuration has the form  $(F' \cdot w(X_j))$  for some  $j$ . In the case of 3, assume  $g(0)$ ,  $E \dot{=} F$ , is the result of BAL using  $F'$ . Without loss of generality, assume it is an application of BAL(L), so  $|F| \leq |F'| + M$ . Suppose the next application of BAL uses  $F''$ . BAL(L) is permitted provided the side condition holds, but BAL(R) is not permitted until a bottom of the application of BAL(L) occurs and the side condition holds. By part 2 a bottom will occur within  $M^2 + M$  steps. Therefore, if BAL(L) is not applied, the goal  $g(i)$  with a bottom has the form  $(F' \cdot w(X_j)) \dot{=} F_i$  and  $|F_i| \leq |F'| + M^2 + 2M$  and  $|(F' \cdot w(X_j))| \leq |F'| + M$ . Assume that the next BAL is BAL(L). If  $F''$  is  $F'$ , or occurs above  $F'$ , then the result follows. Moreover, if it is below  $F'$ , then using part 1 above the result follows. Otherwise, the next BAL is a BAL(R) and so  $F''$  must be  $(F' \cdot w(X_j))$  or occur below it, in which case the result also follows. Part 4 extends part 3. Assume  $g(0)$  is a result of BAL(L) using  $F'$  and the next application of BAL is  $g(k)$  using  $F''$  and  $k \geq M^3 + 3M^2 + 3M$ . Within  $M^2 + M$  steps there is a goal  $g(i)$  with a bottom  $(F' \cdot w(X_j)) \dot{=} F_i$  and  $|F_i| \leq |F'| + M^2 + 2M$  and  $(F' \cdot w(X_j)) \leq |F'| + M$ . By part 1, if there is no application of BAL within a further  $M^3 + 2M^2 + M$  steps then there are goals  $g(i)$ ,  $E_i \dot{=} F'_i$  and  $|F'_i| < |F'|$ , and  $g(j)$ ,  $E'_j \dot{=} F'_j$  and  $|E'_j| < |F'|$ . The result now follows.  $\square$

**Lemma 1** *Assume that  $g(0), \dots, g(n)$  is a sequence of goals in a branch and  $g(0)$  is  $E \dot{=} F$  and there are no applications of BAL in this branch. If  $|E|, |F| \leq k$ , and  $n \geq f_1(k)$ , then the branch contains a successful final goal.*

**Proof:** Assume that  $g(0), \dots, g(n)$  is a sequence of goals in a branch and  $g(0)$  is  $E \dot{=} F$  and  $|E|, |F| \leq k$  and there are no applications of BAL in this branch. By proposition 1.2 within at most  $M^2 + M$  steps both BAL rules are permitted provided that their side conditions hold and at that stage the goal  $E' \dot{=} F'$  has the property that  $|E'|, |F'| \leq (k + M^2 + M)$ . Using proposition 1.1, within a further  $M(k + M^2 + M)$  steps there are goals  $g(i), E_i \dot{=} F_i$  where  $|E_i| = 1$ , and  $g(j), E_j \dot{=} F_j$  where  $|F_j| = 1$ . Without loss of generality assume  $j \geq i$ . Within any interval  $g(j') \dots g(j' + M)$ ,  $j' \geq j$ , there must be goals  $E_{j''} \dot{=} F_{j''}$  with  $|E_{j''}| = 1$  and  $E_{j''} \dot{=} F_{j''}$  with  $|F_{j''}| = 1$ : otherwise, BAL can apply. Therefore, every goal  $g(j'), E_{j'} \dot{=} F_{j'}, j' \geq j$ , has the property  $|E_{j'}| \leq M$  and  $|F_{j'}| \leq M$ . Therefore, within goal(M) steps there must be a repeat goal, so there must be a successful final goal. Overall, within  $f_1(k)$  steps, see definition 1 of section 4.4, there is a successful final goal.  $\square$

Lemma 1 establishes termination of the decision procedure for a sufficiently long branch of goals that does not involve applications of BAL. An analysis is now developed for branches that involve repeated applications of BAL.

**Definition 1** Assume  $F_0, F_1, \dots, F_t$  are successive configurations used in applications of BAL in a branch and assume words  $u_1, \dots, u_t$  such that  $F_i = (F_0 \cdot u_1 \dots u_i)$ . Let  $F_i = \beta_1 G_1 + \dots + \beta_n G_n$  be in  $m$ -head form,  $m \geq 1$ .  $F_i$  is  $m$ -low in the interval  $F_j F_{j+1} \dots F_{j+k}$ ,  $j \geq i$  and  $j + k \leq t$ , if there is a prefix  $v$  of  $u_{j+1} \dots u_{j+k}$  such that  $(F_i \cdot u_{i+1} \dots u_j v) = G_l$  for some  $l$ , and  $F_i$  is then said to be  $m$ -low with  $G_l$  using  $v$  in this interval. The definition is extended to 0-low:  $F_i$  is 0-low with  $F_i$  using  $\varepsilon$  in the interval  $F_j F_{j+1} \dots F_{j+k}$  provided that  $j = i$  and  $F_i$  is not  $m$ -low for any  $m > 0$  in this interval.

**Proposition 2** Assume  $F_0, F_1, \dots, F_t$  are successive configurations used in applications of BAL in a branch and assume words  $u_1, \dots, u_t$  such that  $F_i = (F_0 \cdot u_1 \dots u_i)$ . Assume  $F_i$  is  $m_1$ -low in  $F_j F_{j+1}$ ,  $m_1 \geq 0$  and  $j \geq i$  with  $G$  using  $v$ , and  $F_i$  is not  $m'_1$ -low for any  $m'_1 > m_1$  in  $F_j F_{j+1} \dots F_t$ . Assume  $F_{j+1}$  is  $m_2$ -low in  $F_l F_{l+1}$ ,  $j + 1 \leq l < t$ , and  $m_2 \geq 0$ , with  $G'$  and  $F_{j+1}$  is not  $m'_2$ -low in  $F_l F_{l+1}$  for any  $m'_2 > m_2$ .

1. If  $G = X_1 H_1 + \dots + X_k H_k$  is in 1-head form and  $u_{j+1} = vv'$ , then for all prefixes  $w$  of  $v' u_{j+1} \dots u_t$ ,  $(G \cdot w) = (X_1 \cdot w) H_1 + \dots + (X_k \cdot w) H_k$ .
2.  $|F_{j+1}| \leq |G| + (M^2 + 2M)$  and  $m_2 \leq (M^2 + 2M)$ .
3. If  $G = \beta_1 G_1 + \dots + \beta_n G_n$  is in  $M + 1$ -head form, then the goal that is the result of BAL using  $F_{j+1}$  has the head/tail form  $E_1 G_1 + \dots + E_n G_n \dot{=} F_1 G_1 + \dots + F_n G_n$ , where  $|E_i|, |F_i| \leq M^2 + 5M + 2$ .
4. If  $G = \beta_1 G_1 + \dots + \beta_n G_n$  and  $G' = \delta_1 G'_1 + \dots + \delta_{n'} G'_{n'}$  are in  $M + 1$ -head form, then in these forms  $G'$  is an extension of  $G$  with extension  $e$  and  $|e| \leq M^2 + 2M$ .

**Proof:** For part 1, if there is a prefix  $w$  such that  $(G \cdot w) = H_j$ , then  $F_i$  is  $(m_1 + 1)$ -low in  $F_j \dots F_t$ , contrary to assumption. Part 2 follows from proposition 1.3 and the observation that  $|F_j| \geq |G|$ . For part 3, assume without loss of generality that it is an application of BAL(L) that uses  $F_{j+1}$ , and the resulting goal is  $E_1 (F_{j+1} \cdot w(X_1)) + \dots + E_k (F_{j+1} \cdot w(X_k)) \dot{=} F'$ . Using parts 1 and 2,  $F_{j+1} = (\beta_1 \cdot v') G_1 + \dots + (\beta_n \cdot v') G_n$  and  $|(\beta_i \cdot v')| \leq M^2 + 3M + 1$  and so  $|E_i (F_{j+1} \cdot w(X_i))| \leq M^2 + 5M + 2$  and  $|F'| \leq M^2 + 4M + 1$ . Part 4 follows from parts 1 and 2.  $\square$

**Lemma 2** Assume that there is a subsequence of goals in a branch  $g(i), E_1^i G_1^i + \dots + E_{n_i}^i G_{n_i}^i \dot{=} F_1^i G_1^i + \dots + F_{n_i}^i G_{n_i}^i$ ,  $i : 0 \leq i \leq t$  and each  $g(j+1)$  extends  $g(j)$  by  $e_j$

where  $|e_j| \leq d$  and each head  $|E_j^i|, |F_j^i| \leq h$ . If  $w = \text{width}(h)$  and  $t \geq f_2[d, h, w](w)$ , then there is a successful final goal in the branch.

**Proof:** It is shown that the extension theorem must apply in this branch, so there is a successful final goal. Each head  $|E_j^i|, |F_j^i|$  is bounded by  $h$ . Therefore, the number of goals with different heads is bounded by  $\text{goal}(h)$  and the maximum width of a goal,  $n_i$ , is bounded by  $w = \text{width}(h)$ . Moreover, the number of different extensions,  $e_j$  where  $|e_j| \leq d$ , is also bounded by  $\text{ext}(d, w)$ . Therefore, within  $g(0), \dots, g(f_2[d, h, w](0))$  there are two goals with the same heads because  $f_2[d, h, w](0) = \text{goal}(h)$ , and the second goal is an extension of the first with extension bounded by  $\text{goal}(h) \times d$ . Therefore, within  $g(0), \dots, g(f_2[d, h, w](1))$  there must be four goals with the same heads,  $g'(i)$ ,  $E_1G_1^i + \dots + E_nG_n^i \doteq F_1G_1^i + \dots + F_nG_n^i$ ,  $1 \leq i \leq 2$ , and  $h'(i)$ ,  $E_1H_1^i + \dots + E_nH_n^i \doteq F_1H_1^i + \dots + F_nH_n^i$ ,  $1 \leq i \leq 2$ , and  $g'(2)$  extends  $g'(1)$  by  $e_1$  and  $h'(2)$  extends  $h'(1)$  by  $e_1$  and  $|e_1| \leq \text{goal}(h) \times d$ . The argument is now iterated. Within  $g(0), \dots, g(f_2[d, h, w](w))$  there are  $2^w$  goals  $g'(i)$  and  $2^w$  goals  $h'(i)$  that jointly obey the extension theorem.  $\square$

Using the above results, part 2 of theorem 2 and theorem 3 of section 4.4 are now proved.

### Theorem 2

1. If  $E \sim F$ , then the decision procedure terminates with “successful” tableau.
2. If  $|E|, |F| < n$ , then the decision procedure with root  $E \doteq F$  terminates within  $f(n)$  steps.

**Proof:** Assume that  $E \sim F$ . The tableau for  $E \doteq F$  is built using the rules of the proof system. By completeness of rule application at each stage each frontier goal is true. Therefore, it is not possible to reach an unsuccessful final goal. It is also not possible to become stuck because UNF can always apply. Hence the only issue is that the decision procedure doesn't terminate. Assume that there is an infinite branch of goals  $g(0), \dots, g(n), \dots$  where  $g(0)$  is the root goal  $E \doteq F$  that does not contain a successful final goal. If there are only finitely many applications of BAL, then there is an infinite suffix of goals  $g(i), \dots, g(n), \dots$  and there are no applications of BAL. However, by lemma 1 there must be a successful final goal. Otherwise there are infinitely many applications of BAL. Let  $F_0, F_1 \dots$  be the successive configurations used in applications of BAL. Start with  $F_0$  and find the largest  $m_0$  and the first interval  $F_{i_1-1}F_{i_1}$  such that  $F_0$  is  $m_0$ -low in this interval. By assumption,  $F_0$  is not  $m'$ -low, for any  $m' > m_0$  in any later interval. Next, consider  $F_{i_1}$ . Find the largest  $m_1$  and the first interval  $F_{i_2-1}F_{i_2}$ ,  $i_2 > i_1$ , such that  $F_1$  is  $m_1$ -low in this interval. Using proposition 2, the application of BAL using  $F_{i_1}$ , the goal  $g(k_1)$ , has a head/tail form  $E_1^1G_1^1 + \dots + E_{n_1}^1G_{n_1}^1 \doteq F_1^1G_1^1 + \dots + F_{n_1}^1G_{n_1}^1$  and the application of BAL using  $F_{i_2}$ , the goal  $g(k_2)$ , has head/tail form  $E_1^2G_1^2 + \dots + E_{n_2}^2G_{n_2}^2 \doteq F_1^2G_1^2 + \dots + F_{n_2}^2G_{n_2}^2$  where  $|E_j^i|, |F_j^i| \leq h = M^2 + 5M + 2$  and  $g(k_2)$  extends  $g(k_1)$  by  $e_1$  where  $|e_1| \leq d = M^2 + 2M$ . The argument is now repeated giving an infinite subsequence of goals  $g(k_1), g(k_2), \dots, g(k_n), \dots$  such that each  $g(k_{i+1})$  is an extension of  $g(k_i)$  by  $e_i$  with  $|e_i| \leq d$ , and the heads have bounded size no more than  $h$ . Therefore, by lemma 2 there is a successful final goal within this subsequence.

For part 2, the argument above is refined. Assume a branch  $g(0), \dots, g(t)$  where  $g(0)$  is the root  $E \doteq F$ , and  $|E|, |F| \leq n$  and  $t \geq f(n)$ . If there are no applications of BAL, then by lemma 1 there is a successful final goal within  $f_1(n)$  steps and  $f_1(n) \leq f(n)$ . Let  $F_0, F_1, \dots, F_l$  be successive configurations used in applications of BAL. Let  $d$  and  $h$  be as defined above and let  $w = \text{width}(h)$  and let  $b = f_2[d, h, w](w)$ . Via proposition 2, if no  $F_i$  has a low point that is greater than 0, then for  $l \leq b$  the goals that are the result of these applications of BAL obey lemma 2. By proposition 1,

$|F_{i+1}| \leq |F_i| + d$ . Hence, it follows that the largest  $F_i$  is bounded by  $u = n + bd$ . Moreover, it follows that the maximum number of steps between  $F_i$  and  $F_{i+1}$  is at most  $f_1(u)$  (because, otherwise there must be a successful final goal using lemma 1). For the general case let  $f'(b)$  be the number of successive configurations used in applications of BAL such that there is a subsequence of length  $b$  such that the goals that are the result of these applications of BAL obey lemma 2.  $F_0$  may have  $n$  low points. Therefore,  $f'(b) = n \times f'(b - 1)$ . Subsequent  $F_i$  have only  $d$  low points, so  $f'(b - 1) = d \times f'(b - 2)$ . The overall bound is therefore  $n \times d^b$ . The maximum number of steps between  $F_i$  and  $F_{i+1}$  is again at most  $f_1(u)$ . Therefore, within  $f(n)$  steps it follows that the decision procedure must terminate.  $\square$

## 6 Higher-Order Pushdown Automata

to be added.

### References

- [1] Alur, R. and Madhusudan P. (2004). Visibly pushdown languages. *Procs. STOC 2004*, 202-211.
- [2] Baeten, J., Bergstra, J., and Klop, J. (1993). Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of ACM*, **40**, 653-682.
- [3] Büchi, J. (1962). On a decision method in restricted second order arithmetic. *Logic, Methodology and Philosophy of Science*, Proc. 1960 congress, Stanford Univ. Press, Stanford, CA, 1-11.
- [4] Burkart, O., Caucal, D., and Steffen, B. (1995). An elementary bisimulation decision procedure for arbitrary context-free processes. *Lecture Notes in Computer Science*, **969**, 423-433.
- [5] Burkart, O., and Steffen, B. (1994). Pushdown processes: parallel composition and model checking. *Lecture Notes in Computer Science*, **836**, 98-113.
- [6] Burkart, O., Caucal, D. Moller, F., and Steffen, B. (2001). Verification on infinite structures. In *Handbook of Process Algebra*, ed. Bergstra, J., Ponse, A., and Smolka, S., 545-623, North-Holland.
- [7] Caucal, D. (1992). On the regular structure of prefix rewriting. *Theoretical Computer Science*, **106**, 61-86.
- [8] Caucal, D. (1995). Bisimulation of context-free grammars and of pushdown automata. *CSLI Lecture Notes*, **53**, 85-106.
- [9] Caucal, D. (1996). On infinite transition graphs having a decidable monadic theory. *Lecture Notes in Computer Science*, **1099**, 194-205.
- [10] Christensen, S., Hirshfeld, Y. and Moller, F. (1993). Bisimulation equivalence is decidable for all basic parallel processes. *Lecture Notes in Computer Science*, **715**, 143-157.
- [11] Christensen, S., Hüttel, H., and Stirling, C. (1995). Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, **121**, 143-148.
- [12] Ginsberg, S., and Greibach, S. (1966). Deterministic context-free languages. *Information and Control*, 620-648.
- [13] Groote, J., and Hüttel, H. (1994). Undecidable equivalences for basic process algebra. *Information and Computation*, **115**, 354-371.
- [14] Harrison, M. (1978). *Introduction to Formal Language Theory*, Addison-Wesley.
- [15] Harrison, M., and Havel, I. (1973). Strict deterministic grammars. *Journal of Computer and System Sciences*, **7**, 237-277.
- [16] Harrison, M., Havel, I., and Yehudai, A. (1979). On equivalence of grammars through transformation trees. *Theoretical Computer Science*, **9**, 173-205.

- [17] Hopcroft, J., and Ullman, J. (1979). *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley.
- [18] Hüttel, H., and Stirling, C. (1991). Actions speak louder than words: proving bisimilarity for context free processes. *Proceedings 6th Annual Symposium on Logic in Computer Science*, IEEE Computer Science Press, 376-386.
- [19] Jancar, P. (1997). Decidability of bisimilarity for one-counter processes. *Lecture Notes in Computer Science*, **1256**, 549-559.
- [20] Korenjak, A and Hopcroft, J. (1966). Simple deterministic languages. *Procs. 7th Annual IEEE Symposium on Switching and Automata Theory*, 36-46.
- [21] Muller, D., and Schupp, P. (1985). The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, **37**, 51-75.
- [22] Oyamaguchi, M., Honda, N., and Inagaki, Y. (1980). The equivalence problem for real-time strict deterministic languages. *Information and Control*, **45**, 90-115.
- [23] Rabin, M. (1969). Decidability of second-order theories and automata on infinite trees. *Transactions of American Mathematical Society*, **141**, 1-35.
- [24] Sénizergues, G. (1997). The equivalence problem for deterministic pushdown automata is decidable. *Lecture Notes in Computer Science*, **1256**, 671-681.
- [25] Sénizergues, G. (1998). Decidability of bisimulation equivalence for equational graphs of finite out-degree. *Procs. IEEE 39th FOCS*, 120-129.
- [26] Sénizergues, G. (2001).  $L(A) = L(B)$ ? decidability results from complete formal systems. *Theoretical Computer Science*, **251**, 1-166.
- [27] Sénizergues, G. (2002).  $L(A) = L(B)$ ? a simplified decidability proof. *Theoretical Computer Science*, **281**, 555-608.
- [28] Srba, J. (2002). Undecidability of weak bisimilarity for pushdown processes. *Lecture Notes in Computer Science*, **2421**, 579-593.
- [29] Stirling, C. (2001). Decidability of DPDA equivalence. *Theoretical Computer Science*, **255**, 1-31.
- [30] Stirling, C. (2002) Deciding DPDA equivalence is primitive recursive. *Lecture Notes in Computer Science*, **2380**, 821-832.