

Deciding DPDA Equivalence is Primitive Recursive

Colin Stirling
Division of Informatics
University of Edinburgh

email: cps@dcs.ed.ac.uk

January 2002

Abstract

A deterministic decision procedure for deciding equivalence of deterministic context-free languages is presented. An upper bound on the complexity of the procedure is given that is primitive recursive.

1 Introduction

The DPDA equivalence problem was posed in 1966 [4]: is there an effective procedure for deciding whether two configurations of a deterministic pushdown automaton (a DPDA) accept the same language? The problem is whether language equivalence is decidable for deterministic context-free languages. Despite intensive work throughout the late 1960s and 1970s, the problem remained unsolved until 1997 when Sénizergues announced a positive solution [12]. 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. The full proof by Sénizergues, in journal form, appeared in [13]. It exposes structure within a DPDA by representing configurations as boolean rational series, and he develops an algebraic theory of their linear combinations. Equivalence between configurations is captured within a deduction system. The equations within the proof system have associated weights. Higher level strategies (transformations) are defined which guide proof. A novel feature is that these strategies depend upon differences between weights of their associated equations. Decidability is achieved by showing that two configurations are equivalent if, and only if, there is a finite proof of this fact.

I produced a different proof of decidability that is essentially a simplification of Sénizergues’s proof [16], and see [14]. It is based on a mixture of techniques developed in concurrency theory and language theory. The first step is to view the DPDA problem as a bisimulation equivalence problem for a process calculus whose expressions generate infinite state transition systems. The process calculus is built from determinising strict grammars: strict grammars were introduced by Harrison and Havel [5] because they are equivalent to DPDA. Tableaux proof systems have been used to show decidability of bisimulation equivalence between infinite state processes, see, for instance, [9, 2]. I use this method for the DPDA problem. However, the tableau proof system uses conditional proof rules that involve distances between premises. Essentially this is Sénizergues’s use of weights, and the idea was developed from trying to understand his proof.

The proof of decidability is very complex because the proof of termination uses a mechanism for “decomposition” that in [16] is based on unifiers and auxiliary recursive nonterminals (from [3, 15]). Sénizergues uses a more intricate mechanism. This means that the syntax of the starting process calculus has to be extended in [16] with auxiliary symbols. It also introduces nondeterminism into tableaux with the consequence that the decision procedure (in both [13, 16]) is two semi-decision procedures. The result is that there is no known upper bound on complexity. This is also the case with proofs of decidability in restricted cases, such as for DPDA without ε -transitions that was first shown in 1980, [11].

In this paper I describe a simpler decision procedure that is deterministic and that avoids the decomposition mechanism for termination (the rule CUT in [16] and the transformation T_C in [13]). Instead, there is a new and simpler analysis of termination, centred on a combinatorial result, “the extension theorem”. One consequence is that the syntax of the starting process calculus is not extended. Another consequence is a primitive recursive upper bound on the complexity of the procedure. Section 2 introduces the DPDA problem and Section 3 recasts it as a bisimulation equivalence problem for a particular process calculus. Section 4 describes some features of the process calculus in more detail. The deterministic decision procedure is described in Section 5. Section 6 contains a proof of the extension theorem and Section 7 contains proofs of the correctness of the procedure and of the complexity upper bound.

2 Deterministic pushdown automata

A deterministic pushdown automaton, a DPDA, consists of finite sets of states P , stack symbols S , terminals A and basic transitions T . A basic transition is $pS \xrightarrow{a} q\alpha$ where p, q are states in P , $a \in A \cup \{\varepsilon\}$, S is a stack symbol in S and α is a sequence of stack symbols in S^* . Basic transitions are restricted.

if $pS \xrightarrow{a} q\alpha \in T$ and $pS \xrightarrow{a} r\beta \in T$ and $a \in A \cup \{\varepsilon\}$, then $q = r$ and $\alpha = \beta$
if $pS \xrightarrow{\varepsilon} q\alpha \in T$ and $pS \xrightarrow{a} r\beta \in T$ then $a = \varepsilon$

A configuration of a DPDA has the form $p\delta$ where $p \in \mathbf{P}$ is a state and $\delta \in \mathbf{S}^*$ is a sequence of stack symbols. The transitions of a configuration are determined by the following prefix rule, assuming that $\beta \in \mathbf{S}^*$.

$$\text{if } pS \xrightarrow{a} q\alpha \in \mathbf{T}, \text{ then } pS\beta \xrightarrow{a} q\alpha\beta$$

On input a , the configuration $pS\beta$ in state p with S at the top of the stack changes to state q and α replaces S , so the new configuration is $q\alpha\beta$. Alternatively, with respect to a generational, or process calculus, perspective the configuration $pS\beta$ generates, or performs, a and becomes $q\alpha\beta$. In both these accounts ϵ -transitions have a special status. If a is ϵ then configuration $pS\beta$ may change to $q\alpha\beta$ without reading an input, or $pS\beta$ may silently evolve to $q\alpha\beta$ without performing an observable action in \mathbf{A} . A configuration $p\alpha$ either is “stable” and has no ϵ -transitions or is “unstable” and has a single ϵ -transition (and no other transition).

The transition relation \xrightarrow{a} , $a \in \mathbf{A} \cup \{\epsilon\}$, between configurations is extended to words \xrightarrow{w} , $w \in \mathbf{A}^*$. First, $p\alpha \xrightarrow{\epsilon} p_n\alpha_n$, if $p_n = p$ and $\alpha_n = \alpha$ or there is a sequence of transitions $p\alpha \xrightarrow{\epsilon} p_1\alpha_1 \xrightarrow{\epsilon} \dots \xrightarrow{\epsilon} p_n\alpha_n$. If $w = av \in \mathbf{A}^+$, then $p\alpha \xrightarrow{w} q\beta$ if $p\alpha \xrightarrow{\epsilon} p'\alpha' \xrightarrow{a} q'\beta' \xrightarrow{v} q\beta$.

Fact 1 *If $p\alpha \xrightarrow{w} q\beta$ and $p\alpha \xrightarrow{w} r\delta$ and $q\beta$ and $r\delta$ are stable configurations, then $q = r$ and $\beta = \delta$.*

The language accepted, or generated, by a configuration $p\delta$, $L(p\delta)$, is the set of words $\{w \in \mathbf{A}^* : \exists q \in \mathbf{P}. p\delta \xrightarrow{w} q\epsilon\}$. Acceptance is by empty stack and not by final state. A configuration $p\alpha$ is redundant, if $L(p\alpha) = \emptyset$.

The DPDA problem is whether $L(p\alpha) = L(q\beta)$. Clearly, it is sufficient to restrict the problem to stable configurations that are not redundant. Moreover, one can assume that the DPDA is in normal form that obeys the following constraints.

- if $pS \xrightarrow{a} q\alpha \in \mathbf{T}$, then the length of α , $|\alpha|$, is at most 2
- if $pS \xrightarrow{\epsilon} q\alpha \in \mathbf{T}$, then $\alpha = \epsilon$

Enforcement of the normal form is easy to achieve. In the case of the first constraint, assume that the maximum length of a sequence of stack symbols α in any transition $pX \xrightarrow{a} q\alpha \in \mathbf{T}$ is $n > 2$. New stack symbols $[\beta]$ are introduced to achieve this normal form for sequences of stack symbols β whose length is at most n . The concrete construction is best illustrated by example. If the transitions \mathbf{T} are $pX \xrightarrow{a} pX_1X_2X_3X_4$, $pX_1 \xrightarrow{a} p\epsilon$, $pX_2 \xrightarrow{a} pX_1X_3X_4$, $pX_3 \xrightarrow{a} pX_1X_1X_1X_4$ and $pX_4 \xrightarrow{a} pX_1$, then the largest size of a sequence of stack symbols introduced by a transition is 4. Therefore, extra stack elements $[\beta]$ are introduced whenever β 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, other transitions

and other new stack symbols are introduced: $p[X_2X_3X_4] \xrightarrow{a} p[X_1X_3X_4][X_3X_4]$, $p[X_1X_3X_4] \xrightarrow{a} p[X_3X_4]$, $p[X_3X_4] \xrightarrow{a} p[X_1X_1X_1X_4]X_4$ that introduces a new stack symbol of length 4, $p[X_1X_1X_1X_4] \xrightarrow{a} p[X_1X_1X_4]$, $p[X_1X_1X_4] \xrightarrow{a} p[X_1X_4]$ and $p[X_1X_4] \xrightarrow{a} pX_4$. Transitions for pX_1 and pX_4 are as in \mathbb{T} .

The second constraint is that an ε -transition ‘‘pops’’ the top of the stack. Consider any sequence of ε -transitions from an unstable configuration pX . There are three cases to examine. If $pX \xrightarrow{\varepsilon} p_1X_1\alpha_1 \xrightarrow{\varepsilon} p_2X_2\alpha_2 \xrightarrow{\varepsilon} \dots$, then for any α , $\mathbb{L}(pX\alpha) = \emptyset$. Therefore, the transition $pX \xrightarrow{\varepsilon} p_1X_1\alpha_1 \in \mathbb{T}$ is useless, and can be removed from \mathbb{T} . If $pX \xrightarrow{\varepsilon} p_1X_1\alpha_1 \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} p_nX_n\alpha_n$ and the final configuration is stable, then the initial transition $pX \xrightarrow{\varepsilon} p_1X_1\alpha_1$ is removed from \mathbb{T} and for each transition $p_nX_n \xrightarrow{a} q\beta \in \mathbb{T}$, the transition $pX \xrightarrow{a} q\beta\alpha_n$ is added to \mathbb{T} . If $pX \xrightarrow{\varepsilon} p_1X_1\alpha_1 \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} p_n\varepsilon$, then the initial ε -transition $pX \xrightarrow{\varepsilon} p_1X_1\alpha_1$ is removed from \mathbb{T} and the new transition $pX \xrightarrow{\varepsilon} p_n\varepsilon$ is added to \mathbb{T} .

The motivation for providing a positive solution to the decision question, whether two configurations of a DPDA accept the same language, is to establish decidability of language equivalence between deterministic context-free languages. However, DPDAs which have empty stack acceptance can only recognise the subset of deterministic context-free languages that are prefix-free: a language L is prefix-free if $w \in L$, then no proper prefix of w is also in L . Clearly, if $p\alpha \xrightarrow{w} q\varepsilon$, then it is not possible that $p\alpha \xrightarrow{v} r\varepsilon$ for a proper prefix v of w . However, DPDAs whose acceptance is by final state¹ do recognise all deterministic context-free languages. For any deterministic context-free language L , there is a configuration of a DPDA with empty stack acceptance that accepts the language $\{w\$: w \in L\}$ where $\$$ is a new alphabet symbol, an end marker². Therefore, a decision procedure for language equivalence between configurations of a DPDA with empty stack acceptance establishes decidability of language equivalence between deterministic context-free languages.

Example 1 Let $\mathbb{P} = \{p, r\}$, $\mathbb{S} = \{X, Y\}$ and $\mathbb{A} = \{a, b, c\}$. The basic transitions \mathbb{T} are as follows.

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

This example of a DPDA is in normal form. The configuration $pYYX$ has the

¹A DPDA with acceptance by final state has an extra component $F \subseteq \mathbb{P}$ that is the subset of accepting states: in which case, $\mathbb{L}(p\alpha)$ is the set of words $\{w : p\alpha \xrightarrow{w} q\beta \text{ and } q \in F\}$.

²To 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. Next, for each final state q and stack element S , introduce a basic transition $pS \xrightarrow{\$} e\varepsilon$ where e is a new state whose role is to erase all stack elements, so $eS \xrightarrow{\varepsilon} e\varepsilon$ for all S . Clearly, $p\alpha B \xrightarrow{u} q\beta B$ in the amended DPDA with final state acceptance if, and only if, $p\alpha B \xrightarrow{u\$} e\varepsilon$ in the DPDA that accepts with empty stack acceptance.

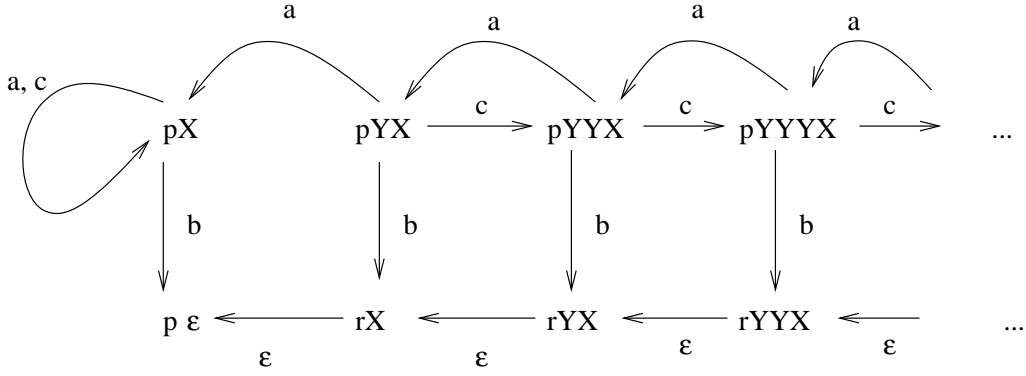


Figure 1: $G(pYX)$

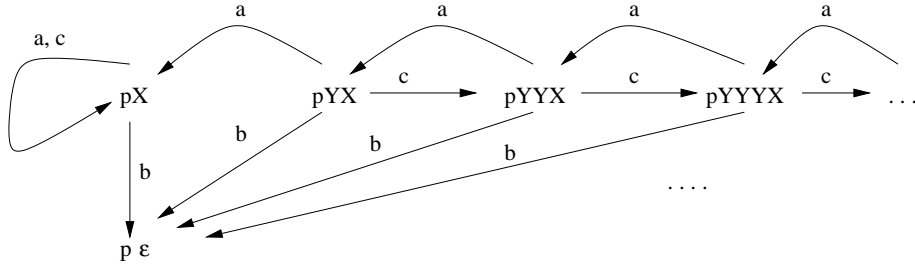


Figure 2: The graph $G^c(pYX)$

c -transition $pYYX \xrightarrow{c} pYYYX$ that is derived from $pY \xrightarrow{c} pYY \in T$ using the prefix rule when $\beta = YX$. \square

If $p\alpha$ is a DPDA configuration, then $G(p\alpha)$ is the possibly infinite state transition graph generated by deriving all possible transitions from $p\alpha$, and deriving all transitions from every reachable configuration from $p\alpha$, using the prefix rule. In $G(p\alpha)$ no special status is accorded to ε -transitions. Figure 1 depicts the transition graph $G(pYX)$ where pYX is a configuration from Example 1. To capture word transitions, the “collapsed” graph of a pushdown automaton, which abstracts from ε -transitions and from redundant configurations, is defined. Consider the stable configurations of Example 1, those of the form $p\alpha$, and transitions \xrightarrow{a} , $a \in A$, between them. This is the collapsed graph, without basic ε -transitions (because there are no redundant configurations). For instance, the collapsed transition graph with stable root pYX , written $G^c(pYX)$, is pictured in Figure 2. In a collapsed graph $p\alpha \xrightarrow{a} q\beta$ if $p\alpha \xrightarrow{a} r\gamma \xrightarrow{\varepsilon} q\beta$. $G^c(p\alpha)$ is possibly an infinite state deterministic transition graph that may have infinite in-degree, as illustrated by the configuration $p\varepsilon$ in Figure 2. If a pushdown automaton does not contain ε -transitions and redundant configurations, then $G^c(p\alpha)$ is the same graph as $G(p\alpha)$ for any of its configurations $p\alpha$. There is a path $p\alpha \xrightarrow{a_1} p_1\alpha_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} p_n\varepsilon$ in $G^c(p\alpha)$ if, and only if, $a_1 \dots a_n \in L(p\alpha)$. Because collapsed graphs are deterministic and do not contain redundant configurations,

language equivalence coincides with bisimulation equivalence.

There is not an obvious relation between the lengths of stacks of equivalent configurations³: in the case of Example 1, $L(pY^n X) = L(pY^m X)$ for every m and n . Techniques for proving decidability of bisimulation equivalence, as developed in the 1990s [1], use decomposition and congruence that allows substitutivity of subexpressions in configurations. However, $L(pY^n) = L(pY^m)$ only if $n = m$. Moreover, the operation of stack extension is not a congruence: if $p\alpha$ and $q\beta$ accept the same language, then this does not imply that $p\alpha\delta$ and $q\beta\delta$ also accept the same language. An important step is to provide a syntactic representation of stable DPDA configurations that dispenses with ε -transitions and that supports congruence, a process calculus that can directly generate transition graphs such as $G^c(pY X)$. The key is *nondeterministic* pushdown automata with a single state and without ε -transitions.

3 Strict deterministic grammars

Strict deterministic grammars were introduced by Harrison and Havel [6], and further studied in [5, 7]. They generate exactly the same languages as DPDA with empty stack acceptance. An attractive feature is that these grammars do not involve ε -productions. However, here they are introduced not as grammars, but as pushdown automata over a single state.

Because the state is redundant, a configuration of a pushdown automaton with a single state is a sequence of stack symbols. Ingredients of such an automaton without ε -transitions, an SDA, are a finite set of stack symbols S , a finite alphabet A and a finite set of basic transitions T . Each basic transition has the form $S \xrightarrow{a} \alpha$ where $a \in A$, S is a stack symbol and α is a sequence of stack symbols.

A configuration of an SDA is a sequence of stack symbols whose transitions are determined by the prefix rule, assuming $\beta \in S^*$: if $S \xrightarrow{a} \alpha \in T$, then $S\beta \xrightarrow{a} \alpha\beta$. The language $L(\alpha)$ accepted, or generated, by a configuration α is the set $\{w \in A^* : \alpha \xrightarrow{w} \epsilon\}$, so acceptance is again by empty stack. Unlike pushdown automata with multiple states, language equivalence is a congruence with respect to stacking: if $L(\alpha) = L(\beta)$, then $L(\alpha\delta) = L(\beta\delta)$. We assume that an SDA is in normal form: if $S \xrightarrow{a} \alpha \in T$, then $|\alpha| \leq 2$ and no element of S is redundant (that is, accepts the language \emptyset).

Any context-free language (different from \emptyset) that does not contain the empty word ε is generable by an SDA, so the decision problem, whether two configurations generate the same language, is undecidable. However, if the SDA is deterministic, then the decision problem is decidable. A deterministic SDA, more commonly known as a “simple grammar”, has restricted basic transitions: if

³A 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, that have no ε -transitions, [11].

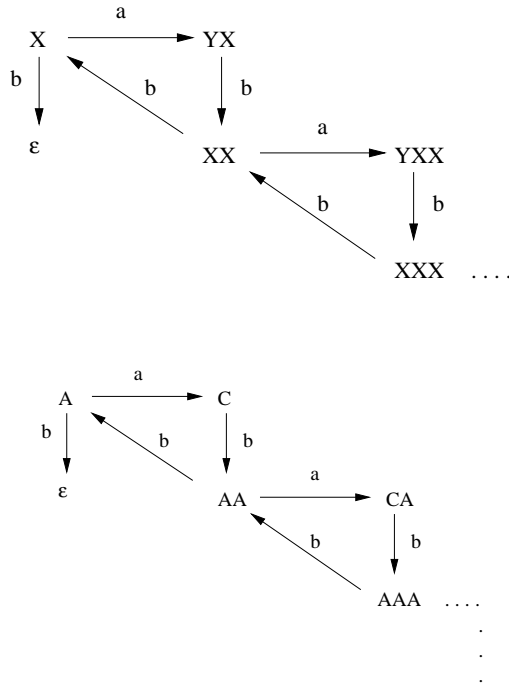


Figure 3: $G(X)$ and $G(A)$

$S \xrightarrow{a} \alpha \in \mathbb{T}$ and $S \xrightarrow{a} \beta \in \mathbb{T}$, then $\alpha = \beta$. Decidability of language equivalence between configurations of deterministic SDA was proved by Korenjak and Hopcroft in 1966 [10]. However, the languages generable by deterministic SDA are strictly contained in the languages generable by DPDA with empty stack acceptance: for instance, $\{a^n b^{n+1} : n > 0\} \cup \{a^n c : n > 0\}$ is not generable by a deterministic SDA.

Instead of assuming determinism, Harrison and Havel included an extra component, \equiv , an equivalence relation on the stack symbols S , in the definition of an SDA. The relation, \equiv , partitions S into disjoint subsets S_1, \dots, S_k , so that for each i , and pair of stack symbols $X, Y \in S_i$, $X \equiv Y$.

Example 1 The following SDA has alphabet $A = \{a, b\}$ and stack symbols $S = \{A, C, X, Y\}$. The partition of S is $\{\{A\}, \{C\}, \{X\}, \{Y\}\}$. The basic transitions \mathbb{T} are as follows.

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

This SDA is deterministic. The transition graphs $G(X)$ and $G(A)$, pictured in Figure 3, are isomorphic, showing how the same transition graph can be generated using different, albeit similar, stack symbols. \square

Example 2 The set $S = \{X, Y, Z\}$, $A = \{a, b, c\}$ and \mathbb{T} is below.

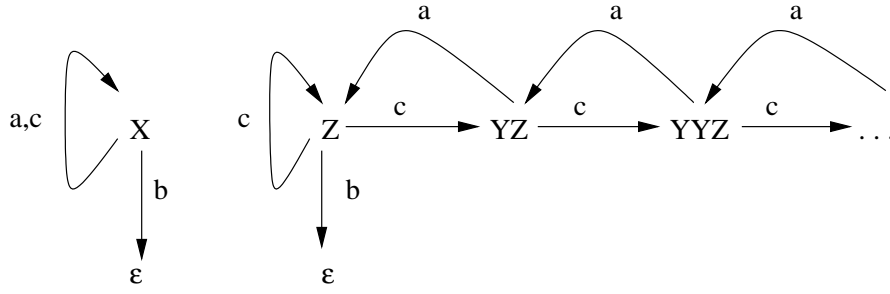


Figure 4: The graph $G(X)$ and $G(Z)$

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

The partition of S is $\{\{X\}, \{Y, Z\}\}$ which means that, for example, $Y \equiv Z$. The graphs of X and Z are illustrated in Figure 4. In particular, $G(Z)$ is nondeterministic because there are two c -transitions from Z . \square

The relation, \equiv , on S is extended to an equivalence relation between sequences of stack symbols, and the same relation, \equiv , is used for the extension.

Definition 1 $\alpha \equiv \beta$ if, either $\alpha = \beta$, or $\alpha = \delta X \alpha'$ and $\beta = \delta Y \beta'$ and $X \equiv Y$ and $X \neq Y$.

Consequently, $\alpha \equiv \beta$ if these sequences are the same or they have a common prefix followed by different stack symbols belonging to the same equivalence class. An instance of equivalent sequences, from Example 2, above, is $XXYY \equiv XXZ$ because $Y \equiv Z$. Some simple properties of \equiv are listed below.

Fact 1

1. $\alpha\beta \equiv \alpha$ if, and only if, $\beta = \epsilon$.
2. $\alpha \equiv \beta$ if, and only if, $\delta\alpha \equiv \delta\beta$.
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 when the following two conditions hold.

1. If $X \equiv Y$ and $X \xrightarrow{a} \alpha$ and $Y \xrightarrow{a} \beta$, then $\alpha \equiv \beta$

2. If $X \equiv Y$ and $X \xrightarrow{a} \alpha$ and $Y \xrightarrow{a} \alpha$, then $X = Y$

An SDA with partition \equiv is strict deterministic (or, just strict) if the relation \equiv on \mathbf{S} is strict⁴. Examples 1 and 2, above, are strict. In the case of Example 1, each partition is a singleton set and hence by condition 1 of Definition 2 this implies determinism. In this extreme case, it follows that $\alpha \equiv \beta$ if, and only if, $\alpha = \beta$ and, therefore, an SDA is then a simple grammar. If the partition involves larger sets, as is the case in Example 2, then constrained nondeterminism is allowed: for example, $Z \xrightarrow{c} Z$ and $Z \xrightarrow{c} YZ$, however, $Z \equiv YZ$ because $Y \equiv Z$.

Some properties of strict SDA are now examined (that are also shown in [7]). First, as the following result implies, the strictness conditions generalise to words $w \in \mathbf{A}^*$.

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 $\alpha \xrightarrow{\varepsilon} \alpha$ and $\beta \xrightarrow{\varepsilon} \beta$ 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 $\beta_1 = \gamma_1\gamma$. Because $X\delta \equiv Y\gamma$ it follows from Fact 1.5 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. First, $\delta_1 = \gamma_1$, and therefore by condition 2 of being strict $X = Y$ and therefore because $X\delta \equiv Y\gamma$ it follows from Fact 1.2 that $\delta \equiv \gamma$ and therefore $\delta_1\delta \equiv \gamma_1\gamma$ from Fact 1.3. Now the required result, $\alpha' \equiv \beta'$, follows by the induction hypothesis because $\alpha_1 \equiv \beta_1$ and $|w'| < |w|$. Second, $\delta_1 \neq \gamma_1$, and therefore because $\delta_1 \equiv \gamma_1$ it now follows from Fact 1.4 that $\delta_1\delta \equiv \gamma_1\gamma$. The required result now follows as in the first case.

The base case for 2 is $|w| = 0$. 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 $\beta_1 = \gamma_1\gamma$. 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 Fact 1.1 it is not possible for $\alpha \equiv \alpha X\lambda$. 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 $X = Y$. \square

⁴More generally, an SDA without a partition is strict deterministic if there exists a strict partition of its stack symbols. Harrison and Havel show that it is decidable (in polynomial time) whether an SDA is strict deterministic [6].

The next result shows that if $\alpha \equiv \beta$ then their languages are prefix disjoint and also, if $\alpha \neq \beta$ then their languages are disjoint.

Proposition 2

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

Proof: To show 1 assume that $\alpha \xrightarrow{w} \epsilon$ and $\alpha \equiv \beta$. If $\beta \xrightarrow{w} \gamma$ then by Proposition 1.1 $\epsilon \equiv \gamma$, so, by Proposition 1.2 $\alpha = \beta$. Therefore, it is not possible that $\beta \xrightarrow{wav} \epsilon$. Part 2 is now an immediate corollary. \square

An alternative characterisation of the collapsed graphs of DPDA is developed using strict SDA. A configuration of a strict SDA is a sequence of stack symbols. The definition of configuration is extended to sets of sequences of stack symbols, $\{\alpha_1, \dots, \alpha_n\}$, written in sum form $\alpha_1 + \dots + \alpha_n$. Two sum configurations are equal, written using $=$, if they are the same set. A degenerate case is the empty sum, written \emptyset . The language of a sum configuration is the union of the languages of the components: $\mathbf{L}(\alpha_1 + \dots + \alpha_n) = \bigcup \{\mathbf{L}(\alpha_i) : 1 \leq i \leq n\}$. Therefore, $\mathbf{L}(\emptyset) = \emptyset$.

Only a subset of sum configurations are interesting.

Definition 3 A sum configuration $\beta_1 + \dots + \beta_n$ is *admissible*, if $\beta_i \equiv \beta_j$ for each pair of components, and $\beta_i \neq \beta_j$ when $i \neq j$.

The empty sum, \emptyset , is therefore admissible. In [7] admissible configurations are called ‘‘associates’’. Some example admissible configurations of Example 2, above, are XX , $ZZZ + ZZY$, $YX + Z$, $Z + YZ$ and $Z + YZ + YYZ$. An example of a configuration that is not admissible is $X + Y$ because $X \not\equiv Y$. A simple corollary of Proposition 1 is that admissibility is preserved by word transitions.

Fact 2 If $\{\beta_1, \dots, \beta_n\}$ is admissible, then $\{\beta' : \beta_i \xrightarrow{w} \beta', 1 \leq i \leq n\}$ is admissible, for any word $w \in \mathbf{A}^*$.

A strict SDA can be determined, by determining the basic transitions \mathbf{T} to \mathbf{T}^d . For each stack symbol X and $a \in \mathbf{A}$, the family of transitions $X \xrightarrow{a} \alpha_1, \dots, X \xrightarrow{a} \alpha_n$ in \mathbf{T} is changed to the single transition $X \xrightarrow{a} \alpha_1 + \dots + \alpha_n$ in \mathbf{T}^d . (The emptysum configuration is \emptyset .) The sum configuration $\alpha_1 + \dots + \alpha_n$ is admissible. The prefix rule for generating transitions is also extended to admissible configurations.

- If $X_1\beta_1 + \dots + X_m\beta_m$ is admissible and $X_i \xrightarrow{a} \sum \alpha_{ij} \in \mathbf{T}^d$ for each i , then $X_1\beta_1 + \dots + X_m\beta_m \xrightarrow{a} \sum \alpha_{1j}\beta_1 + \dots + \sum \alpha_{mj}\beta_m$

By Fact 2, the resulting configuration is admissible.

Given a determined strict SDA and an admissible configuration $\alpha_1 + \dots + \alpha_n$, the graph $\mathbf{G}^d(\alpha_1 + \dots + \alpha_n)$ is the transition graph generated by $\alpha_1 + \dots + \alpha_n$, except

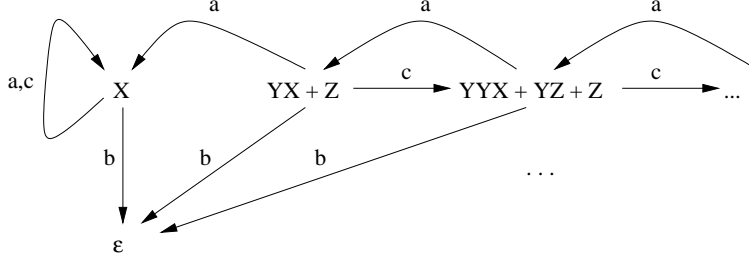


Figure 5: The graph $G^d(YX + Z)$

that useless transitions $\beta_1 + \dots + \beta_k \xrightarrow{a} \emptyset$, involving the empty set configuration, are omitted.

Example 3 In the case of Example 2, above, \mathbb{T}^d is the following set.

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

The graph $G^d(YX + Z)$ is pictured in Figure 5. It is not by accident that, in fact, this graph is similar to the graph of Figure 2. Two example transitions of Figure 5, and their derivation from the extended prefix rule, are listed.

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

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

There is a standard transformation of a pushdown automaton into a language equivalent SDA, or context-free grammar: see, for instance, [5, 8]. Assume a DPDA in normal form with state set \mathbb{P} , stack symbols \mathbb{S} , alphabet \mathbb{A} and transitions \mathbb{T} . An SDA in normal form with stack symbols \mathbb{S}_1 and transitions \mathbb{T}_1 is constructed, in stages, as follows.

1. For every pair of states $p, q \in \mathbb{P}$ and stack symbol $X \in \mathbb{S}$, introduce a stack symbol $[pXq] \in \mathbb{S}_1$.
2. For transitions, the initial step is to define the following whenever $a \in \mathbb{A}$.
 - (a) If $pX \xrightarrow{a} q\epsilon \in \mathbb{T}$, then $[pXq] \xrightarrow{a} \epsilon \in \mathbb{T}_1$.
 - (b) If $pX \xrightarrow{a} qY \in \mathbb{T}$, then $[pXr] \xrightarrow{a} [qYr] \in \mathbb{T}_1$ for each $r \in \mathbb{P}$.
 - (c) If $pX \xrightarrow{a} qYZ \in \mathbb{T}$, then $[pXr] \xrightarrow{a} [qYp'][p'Zr] \in \mathbb{T}_1$ for each r and p' in \mathbb{P} .

3. A stack symbol $[pXq] \in \mathcal{S}_1$ is an ε -symbol, if $pX \xrightarrow{\varepsilon} q\epsilon \in \mathcal{T}$. All ε -symbols are erased from the right hand side of any transition in \mathcal{T}_1 .
4. Next, the SDA is normalised by deleting all redundant stack symbols from \mathcal{S}_1 (and all useless transitions in \mathcal{T}_1 with a redundant stack symbol on its right hand side).

The following result captures essential features of the the SDA that is constructed from a DPDA.

Fact 3

1. If $w \in \mathbf{A}^*$ and pX is stable in the DPDA, then $pX \xrightarrow{w} q\epsilon$ if, and only if, $[pXq] \xrightarrow{w} \varepsilon$ in the SDA.
2. The following relation \equiv on \mathcal{S}_1 is strict: $[pSq] \equiv [rXt]$ if, and only if, $p = r$ and $S = X$.

Harrison and Havel also prove the converse, that any strict SDA can be transformed into a DPDA [6].

The transformation of a DPDA into a strict SDA does not preserve determinism. However, if the SDA is determinised, 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 ε -symbols are erased, and summands involving redundant stack symbols are removed.

Fact 4 $\mathbf{L}(p\alpha) = \mathbf{L}(\text{sum}(p\alpha))$

Moreover, the transition graph $\mathbf{G}^c(p\alpha)$ is almost isomorphic to⁵ $\mathbf{G}^d(\text{sum}(p\alpha))$. There can be $|\mathbf{P}|$ co-roots, vertices of the form $q\varepsilon$, in $\mathbf{G}^c(p\alpha)$ in contrast with the single co-root, ε , in $\mathbf{G}^d(\text{sum}(p\alpha))$.

Example 4 An example is the conversion of the DPDA of Example 1, of the previous section. The stack symbols \mathcal{S}_1 is the set

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

The transitions in \mathcal{T} are then converted.

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

⁵In fact, is bisimulation equivalent to.

There are two ϵ -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 S_1 to the set $\{[pXp], [pYp], [pYr]\}$, and the transitions 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 T_1 are determined: the two c -transitions from $[pYr]$ are replaced by the single transition $[pYr] \xrightarrow{c} [pYr] + [pYp][pYr]$. The resulting strict SDA is Example 2, above, 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 SDA, that generates the same language. \square

The DPDA decision problem is, therefore, equivalent to deciding whether admissible configurations of a determined strict SDA in normal form generate the same language (or, equivalently, are bisimulation equivalent). In the following this problem is solved and a complexity upper bound is given. A DPDA in normal form with stack size s and state size p is transformed into a strict SDA 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 SDA that contain stack sequences whose length is bounded by n .

4 Heads, tails and extensions

Assume a fixed determined strict SDA in normal form with ingredients S , A , T_d and \equiv . Let $<$ be a total ordering on the alphabet A : for each $a, b \in A$, if $a \neq b$ then either $a < b$ or $b < a$. Relative to $<$, a total ordering on words $u \in A^*$ is defined and the same symbol $<$ is employed: $u < v$ if $|u| < |v|$ or $|u| = |v|$ and u is lexicographically smaller⁶ than v . If $u < v$, then u is said to be shorter than v . Assume that α, β, \dots range over sequences of stack symbols. An admissible configuration is a set $\{\beta_1, \dots, \beta_n\}$ where for each i and j , $\beta_i \equiv \beta_j$, that is usually written in sum form $\beta_1 + \dots + \beta_n$. Let E, F, G, \dots range over admissible configurations, and recall that $E = F$ if they are the same set of sequences. Two special cases of sum configurations are \emptyset , the empty set, and ε , the singleton set $\{\varepsilon\}$. The size of an admissible configuration $E = \beta_1 + \dots + \beta_n$, written $|E|$, is the length of its longest sequence, $\max\{|\beta_j| : 1 \leq j \leq n\}$. It is

⁶That is, $u = wau'$ and $v = wbv'$ and, with respect to A , $a < b$.

assumed that $|\emptyset| = 0$. Clearly, for each $n \geq 0$, there are only boundedly many different admissible configurations of size at most n .

A useful notation is “the configuration E after the word u ”, written $E \cdot u$, that is the unique configuration F such that $E \xrightarrow{u} F$, which can be \emptyset . This is unique, and admissible, using Fact 2 of the previous section. Some obvious properties of $E \cdot u$ are listed, below.

Fact 1

1. $(E \cdot \varepsilon) = E$.
2. $(E \cdot uv) = (E \cdot u) \cdot v$.
3. If $(E \cdot u) = \emptyset$, then $(E \cdot uv) = \emptyset$.
4. If $(E \cdot u) = \varepsilon$, then for any $a \in \mathbf{A}$, $(E \cdot ua) = \emptyset$.

Proposition 1

1. $|E \cdot u| \leq |E| + |u|$.
2. If $|\beta| \geq |u|$ and $(\beta \cdot u) \neq \emptyset$, then $|\beta| - |u| \leq |\beta \cdot u| \leq |\beta| + |u|$.

Proof: Because the SDA is in normal form, if $X \xrightarrow{a} E \in \mathbf{T}_d$, then $0 \leq |E| \leq 2$, and, therefore the results follow. \square

The language accepted by configuration E , $\mathbf{L}(E)$, is $\{u : (E \cdot u) = \varepsilon\}$. Two configurations E and F are language equivalent, written $E \sim F$, if they accept the same language, $\mathbf{L}(E) = \mathbf{L}(F)$. Language equivalence can also be approximated. If $n \geq 0$, then E and F are n -equivalent, written $E \sim_n F$, provided that they reject the same words whose length is at most n : for all words w such that $|w| \leq n$, $(E \cdot w) = \emptyset$ if, and only if, $(F \cdot w) = \emptyset$. Given n , it is clearly decidable whether $E \sim_n F$ by exhaustive enumeration.

Fact 2

1. $E \sim F$ if, and only if, for all $n \geq 0$, $E \sim_n F$.
2. If $E \not\sim F$ and $E, F \neq \emptyset$, then there is an $n \geq 0$ such that $E \sim_n F$ and $E \not\sim_{n+1} F$.
3. $E \sim F$ if, and only if, for all $u \in \mathbf{A}^*$, $(E \cdot u) \sim (F \cdot u)$.
4. $E \sim_n F$ if, and only if, for all $u \in \mathbf{A}^*$ where $|u| \leq n$, $(E \cdot u) \sim_{n-|u|} (F \cdot u)$.
5. If $E \sim_n F$ and $0 \leq m < n$, then $E \sim_m F$.
6. If $E \sim_n F$ and $F \not\sim_n G$, then $E \not\sim_n G$.

Definition 1 For each stack symbol X , the word $w(X)$ is the shortest word in the set $\{u : (X \cdot u) = \epsilon\}$. The *norm* of X is $|w(X)|$, the length of $w(X)$.

It is easy to compute $w(X)$ for each stack symbol X . Initially, stack symbols in S with norm 1 are identified by examining transitions in T_d of the form $X \xrightarrow{a} \epsilon$, and their shortest words are defined. There must be at least one stack symbol with norm 1. Next, stack symbols with norm 2 are identified. X has norm 2 if X does not have norm 1 and $X \xrightarrow{a} Z + E \in T_d$ and Z has norm 1. It is possible that there are no such stack symbols. A stack symbol X has norm 3 if it does not have norm 1 or norm 2, and either $X \xrightarrow{a} Z + E \in T_d$ and Z has norm 2 or $X \xrightarrow{a} YZ + E \in T_d$ and both Y and Z have norm 1. Identification of norm is iterated, until $w(X)$ is calculated for each stack symbol X . In the case of example 3 of the previous section each stack symbol has norm 1, $w(X) = b = w(Z)$ and $w(Y) = a$. An important measure of an SDA, in normal form, is its maximum norm.

Definition 2 The maximum norm M of an SDA in normal form whose stack symbols are S is, $\max\{|w(X)| : X \in S\}$.

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

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

Consequently, $w(X_{n-j}) = a^{2^j}$. The notion of norm extends to configurations: $w(E)$ is the shortest word v such that $(E \cdot v) = \epsilon$. (The norm of \emptyset is assumed to be ∞ .)

A feature of the decision procedure is repeating patterns within admissible configurations. An admissible configuration is written in sum form $\beta_1 + \dots + \beta_n$ where each β_i is distinct. The operation $+$ can be extended: if E and F are admissible and $E \cup F$ is admissible and E, F are disjoint, $E \cap F = \emptyset$, then $E + F$ is the admissible configuration $E \cup F$. The operation $+$ on admissible configurations is partial. Sequential composition, written as juxtaposition, is also used: if E and F are admissible, then EF is the configuration $\{\beta\gamma : \beta \in E \text{ and } \gamma \in F\}$, that is admissible using Fact 1.3 of the previous section.

Proposition 2

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

Proof: Parts 1 and 2 are immediate corollaries of Proposition 2 of the previous section, and Part 3 follows from the definition of sequential composition. \square

With respect to language acceptance, the partial operation $+$ on admissible configurations is, therefore, a disjoint union.

Fact 3

1. $E + \emptyset = E = \emptyset + E$
2. $E\emptyset = \emptyset = \emptyset E$
3. $E\varepsilon = E = \varepsilon E$
4. $(E + F)G = EG + FG$
5. $G(E + F) = GE + GF$

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 , $X'_i \equiv X'_j$: see Definition 1 of the previous section. 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 $G_i \neq \emptyset$ is also admissible (by Fact 1 of the previous section). In this presentation, E is in 1-head form. Head form can be generalised.

Definition 3 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 β_1, \dots, β_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 4 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 .

Proposition 3 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 $(\beta_i \cdot u) = \varepsilon$, 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 Proposition 1 of the previous section. $\beta_1 + \dots + \beta_n$ is admissible and $\beta_i \neq \beta_j$ when $i \neq j$. If $(\beta_i \cdot u) = \varepsilon$ and $\beta_j \xrightarrow{u} F$, then $F \equiv \varepsilon$ 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

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

Definition 4 $E = E_1G_1 + \dots + E_nG_n$ 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 the previous section. 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, $YYH_1 + YZH_2 + ZH_3$, where $H_1 = YX + Z$ and $H_2 = H_3 = \varepsilon$. $(E \cdot c)$ is

$$(YY \cdot c)H_1 + (YZ \cdot c)H_2 + (Z \cdot c)H_3 = YYYH_1 + YYZH_2 + (YZ + Z)H_3.$$

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 presented as $E_1G_1 + \dots + E_nG_n$, then assume that it fulfills the conditions of Definition 4 of a head/tail form. The following result lists some properties of head/tail forms. Language 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 4 *Assume $E = E_1G_1 + \dots + E_nG_n$.*

1. *If $(E_i \cdot u) = \varepsilon$, 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 $E_i \neq \varepsilon$ 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 3.1 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 4 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 $E_i \neq \varepsilon$. Therefore, using Proposition 4, 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 tails⁷, then the imbalance between E and F , relative to this presentation, is

⁷Any pair of configurations E and F have a head/tail form involving the same tails: $E = EG$ and $F = FG$ when $G = \varepsilon$.

$\max\{|E_i|, |F_i| : 1 \leq i \leq n\}$. If the imbalance is 0, then they are the same configurations. The next result establishes a key property of a combination of such configurations.

Proposition 5 *Assume the following configurations.*

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

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

Proposition 5 will be particularly useful when the tails H_i are “extensions” of the tails G_i .

Definition 5 If $E = E_1G_1 + \dots + E_nG_n$ and $F = F_1H_1 + \dots + F_mH_m$, then F in its head/tail form is a tail extension of E in its head/tail form provided that each $H_i = K_1^iG_1 + \dots + K_n^iG_n$, $1 \leq i \leq m$. 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 by e .

Extensions are matrices, written in a linear notation, and can be composed, multiplied, as the following result shows.

Proposition 6 *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 $C_t = K_t^i J_1^t G_1 + \dots + K_t^i J_l^t G_l$. Reorganising the expression, $G_i'' = H_1 G_1 + \dots + H_l G_l$ where $H_t = K_1^i J_t^1 + K_2^i J_t^2 + \dots + K_m^i J_t^m$, as required. Clearly $|ef| \leq |e| + |f|$. \square

Example 2 The following uses Example 3 of Section 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

A special case of an extension is when the tails are the same. If $E = E_1 G_1 + \dots + E_n G_n$ and $F = F_1 G_1 + \dots + F_n G_n$, then E extends F by $e = (\varepsilon + \emptyset + \dots + \emptyset, \dots, \emptyset + \emptyset + \dots + \varepsilon)$. This extension e is abbreviated to the identity (ε) .

5 The decision procedure

The procedure for deciding $E \sim F$ is to build a goal directed proof tree, a tableau, with initial goal $E \doteq F$, “is $E \sim F$?”, using proof rules that reduce goals to subgoals. There are just three rules, presented in Figure 6. UNF, for “unfold”, reduces a goal $E \doteq F$ to subgoals $(E \cdot a) \doteq (F \cdot a)$ for each 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} , so $a_1 < a_2 < \dots < a_k$. UNF is complete and sound. If the goal is true, then so are all the subgoals (see Fact 2.3 of the previous section). Soundness is the converse. A finer version, part 2 of Fact 1, uses approximants: if the goal fails at level $m + 1$, then at least one subgoal fails at level m (see Fact 2.4 of the previous section).

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 where X, Y and Z are from Example 3 of Section 3.

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

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

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 there is a word u such that $|u| = m$ and

UNF

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

BAL(R)

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

BAL(L)

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

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 6: The tableau proof rules

$E' = (E \cdot u)$ and $F' = (F \cdot u)$. In this circumstance, u is the word “associated” with the sequence of applications of UNF.

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 $|F'| \leq |F| + m$.*

The other two rules in Figure 6 are conditional that involve two premises: the second premise goal reduces to the subgoal beneath it provided that the first premise is above it (on the path back to the root goal). They are BAL rules, for “balance”, involving substitution of subexpressions, that allow a goal to be reduced to a balanced subgoal where the imbalance between the configurations is bounded.

Example 2 An illustration of an application of BAL(L) uses Example 1 of Section 3, that involves the stack symbols A , C , X and Y .

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

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. There is the same bound on imbalance if the starting goal is $X^n \doteq A^n$, for any $n > 0$. \square

Definition 1 An application of BAL is said to use F if F is the configuration in the initial goal of the rule, as in Figure 6.

Example 2 illustrates how a particular application of BAL bounds imbalance. Proposition 1.2, below, captures precisely the bounds of imbalance: the result is stated for BAL(L), and the symmetric version holds for 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

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

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, $E_1G_1 + \dots + E_nG_n$ 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 $E_1G_1 + \dots + E_nG_n \doteq F_1G_1 + \dots + F_nG_n$ 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. Completeness of an application of BAL is straightforward, part 1 of Proposition 2 below, if the two goals are true, then the subgoal is also true. Soundness of an application of BAL is more intricate. First, “global” soundness of the proof system is explained. If there is a successful tableau whose root is false, then there is a branch of the tableau within which each subgoal is false. The idea is refined using approximants. If the root is false then there is an offending branch (of false goals) in the tableau within which the approximant indices decrease whenever rule UNF has been applied by Fact 1.2, above. Soundness of an application of BAL is that if the two premise goals belong to an offending branch, then the subgoal preserves the level of falsity of the second premise goal.

Proposition 2 [Completeness and soundness of BAL]

1. If $X_1H_1 + \dots + X_kH_k \sim F$ and $E_1H_1 + \dots + E_kH_k \sim F'$, then $E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k)) \sim F'$.
2. If $X_1H_1 + \dots + X_kH_k \sim_{n+m} F$ and $E_1H_1 + \dots + E_kH_k \not\sim_{n+1} F'$ and each $E_i \neq \varepsilon$ and $m \geq \max\{|w(X_i)| : E_i \neq \emptyset\}$, then $E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k)) \not\sim_{n+1} F'$.

Proof: Let E be $X_1H_1 + \dots + X_kH_k$ and assume $E \sim F$. By Proposition 3.1 of the previous section, $(E \cdot w(X_i)) = H_i$, so $H_i \sim (F \cdot w(X_i))$, by Fact 2.3 of the previous section. By assumption of head/tail form, each $H_i \neq \emptyset$ and, therefore, each $(F \cdot w(X_i)) \neq \emptyset$. Consequently, by Proposition 4.5 of the previous section $E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k)) \sim E_1H_1 + \dots + E_kH_k$ 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 by Proposition 4.4 of the previous section, $E_1(F \cdot w(X_1)) + \dots + E_n(F \cdot w(X_n)) \sim_{n+1} E_1H_1 + \dots + E_nH_n$, and now the result follows using Fact 2.6 of the previous section. \square

In Example 2, above, BAL(L) is applied after UNF. However, the other two rules UNF and BAL(R) also apply. 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 7, 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

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

Figure 7: A potential switch from BAL(L) to BAL(R)

strategy is to apply 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 rules⁸. 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 8, and that it is BAL(R), shown on the right in Figure 8. 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

⁸For 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.

$$\begin{array}{ccc}
F & & F \\
\vdots \text{ BAL(L)} & & \vdots \text{ BAL(L)} \\
E_1 \doteq F_1 & & E_1 \doteq F_1 \\
\vdots & & \vdots \\
F' & & (F \cdot w(X_i)) \\
\vdots \text{ BAL(L)} & & \vdots \\
E_2 \doteq F_2 & & F' \\
\vdots & & \vdots \text{ BAL(R)} \\
& & E_2 = F_2 \\
& & \vdots
\end{array}$$

Figure 8: Proof of Proposition 3

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, is below.

$$\frac{\frac{\frac{XX^5 \doteq AA^5}{YXX^5 \doteq CA^5} \text{BAL(L)} \quad X^5 \doteq A^5 \text{ UNF}}{(1) YXA^5 \doteq CA^5} \text{ UNF}}{\emptyset \doteq \emptyset \quad (2) XXA^5 \doteq AA^6 \text{ UNF}} \text{ UNF} \\
\frac{(*) YXXA^5 \doteq CA^6 \quad XA^5 \doteq A^6 \text{ UNF}}{YXA^6 \doteq CA^6} \text{ BAL(L)}$$

At goal (*), BAL(L) is applied. Either of the premises (1) and (2) could be the initial premise for the application: however, by the discussion above it is the lower premise (2). The first application of BAL(L) uses $F_0 = AA^5$ and the second uses $F_1 = AA^6$ and $F_1 = (F_0 \cdot ab)$. \square

Example 4 Below is the initial part of the tableau for Example 1, above.

$$(1) \quad \epsilon \doteq \epsilon \quad \frac{(*) YX + Z \doteq YYX + YZ + Z}{\frac{YYX + YZ + Z \doteq YYYX + YYZ + YZ + Z}{YYYX + YYZ + YZ + Z \doteq YYYX + YYZ + YZ + Z} \text{ UNF}} \text{ BAL(L)}$$

where (1) is the subtableau

$$\begin{array}{c}
\frac{(**) X \dot{=} YX + Z}{X \dot{=} X \quad \epsilon \dot{=} \epsilon} \text{ UNF} \\
\frac{\frac{X \dot{=} YX + Z}{X \dot{=} YYX + YZ + Z} \text{ UNF}}{X \dot{=} YX + Z \quad \epsilon \dot{=} \epsilon} \text{ BAL(R)} \\
\frac{\frac{X \dot{=} YYX + YZ + Z}{X \dot{=} YYYX + YYZ + YZ + Z} \text{ UNF}}{X \dot{=} YYX + YZ + Z} \text{ BAL(R)}
\end{array}$$

The premise (*) is the initial premise for the application of BAL(L), and (**) is the initial premise for the first BAL(R). The leaf goals are either identities or repeats. In fact, it will turn out that this partial tableau is the completed successful tableau that establishes that $L(pYX) = L(pYYX)$ of Example 1 of Section 2. \square

For the tableau construction to be a decision procedure, a notion of final goal is needed so that a tableau can be terminated. The tableau proof rules are locally complete, if a goal is true then so are subgoals. Consequently, if an obviously false subgoal is reached, then the root goal is also false. So the criterion for being an unsuccessful final goal is that it is obviously false. This occurs when the goal has the form $\emptyset \dot{=} E$ or $E \dot{=} \emptyset$ and $E \neq \emptyset^9$. The tableau proof rules are also locally sound, if all the subgoals are true then so is the goal. Therefore, if an obviously true subgoal, $E \dot{=} E$, is reached then it should count as a successful final leaf. However, the tableau proof rules are sound in a finer version. In the case of UNF, if the goal is false at level $m+1$, $E \not\sim_{m+1} F$, then at least one subgoal fails at level m , $(E \cdot a) \not\sim_m (F \cdot a)$. And, as shown above, applications of BAL preserve the falsity index. Consequently, if a subgoal $E \dot{=} F$ is repeated in a branch, and there is at least one application of UNF between them, then the second occurrence of $E \dot{=} F$ can also count as a successful final goal. If the root of the tableau is false, then there is an offending path of false goals in the tableau within which the approximant indices decrease whenever UNF is applied. Consider the branch with $E \dot{=} F$ occurring twice: if this were an offending branch, then at the first occurrence, by Fact 2.2 of the previous section, there is a least $n \geq 0$ such that $E \sim_n F$ and $E \not\sim_{n+1} F$. Therefore, at the second occurrence $E \not\sim_{(n+1)-k} F$ where k is the number of applications of UNF between the two; this is a contradiction when $k \geq 1$.

A repeat is an instance of a more general situation where goals may be growing in size, formally captured below by the ‘‘extension theorem’’. Roughly speaking, in a branch if there are goals where the rates of change of tails are repeating, then there is a successful final goal. A repeat is an instance when the rate of change is zero. In a long enough branch with multiple applications of BAL, by

⁹Clearly, $L(E) \neq L(\emptyset)$ because $w(E) \in L(E)$.

the extension theorem $n = 1$. The families of goals are as follows.

$$\begin{array}{lll} g(1) & YXG^1 \doteq CG^1 & G^1 = A^5 \\ g(2) = h(1) & YXG^2 \doteq CG^2 & G^2 = A^6 \\ h(2) & YXH^2 \doteq CH^2 & H^2 = A^7 \end{array}$$

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 $YXA^7 \sim_m CA^7$. This justifies that the subgoal $YXA^7 \doteq CA^7$ is a successful final goal. The argument is the same as for a repeating goal, above¹⁰.

Definition 3 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 $i \geq 0$

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

The second occurrence of a repeating goal in a branch obeys the extension theorem provided that there is at least one application of UNF between the repeating goals. 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 the first occurrence of the repeating goal, and each extension is the identity, (ϵ) . All three conditions of Definition 3 are thereby satisfied.

Definition 4 Assume a branch of goals $g(0), \dots, g(n)$ where $g(0)$ is the root goal. The goal $g(n)$ is a final goal in the following circumstances.

1. If $g(n)$ is an identity $E \doteq E$, then $g(n)$ is a successful final goal

¹⁰Assume 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 $h(2)$.

2. If $g(n)$ obeys the extension theorem, then $g(n)$ is a successful final goal
3. If $g(n)$ has the form $E \doteq \emptyset$ or $\emptyset \doteq E$ and $E \neq \emptyset$, then $g(n)$ is an unsuccessful final goal.

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.

The main results of the paper are the following that prove decidability of DPDA equivalence and establish a complexity upper bound that is primitive recursive. Their proofs are presented in Section 7.

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”.

From Section 3, a DPDA with s_1 stack symbols and p states is transformed into an SDA 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 SDA 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 SDA 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, $\text{ext}(d, w) \leq 2^{s^{dw}}$). 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 5

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 2 *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 3 *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, SDA) 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. A special case is simple grammars for which there is a polynomial time algorithm, see [1]. In this case the extension theorem is not necessary because $\alpha\delta \sim \beta\delta$ if, and only, if $\alpha \sim \delta$, and, therefore, after an application of BAL the common tail can be cut. The analysis in this paper using the extension theorem is exponential, and not optimal. However, the result here, effectively, covers bisimulation equivalence of “unnormalized” simple grammars (that may involve redundant stack symbols), and it is not known if it is decidable in polynomial time. In the unnormalized case, if $\alpha\delta \sim \beta\delta$, then this does not imply that $\alpha \sim \delta$. However, the extension theorem does hold.

6 Proof of the extension theorem

The extension theorem, Theorem 1 of the previous section, 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) $G_i^s \sim_m J_1 G_1^s + \dots + J_n G_n^s$ 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 $E_1 G_1^i + \dots + E_n G_n^i \doteq F_1 G_1^i + \dots + F_n G_n^i$ and each goal $h(i)$ has the form $E_1 H_1^i + \dots + E_n H_n^i \doteq F_1 H_1^i + \dots + F_n H_n^i$, 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 $g(1)$, $E_1 G_1 + \dots + E_n G_n \doteq F_1 G_1 + \dots + F_n G_n$, abbreviated to $E \doteq F$, and $h(1)$, $E_1 H_1 + \dots + E_n H_n \doteq F_1 H_1 + \dots + F_n H_n$, 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 $G_i \sim_m J_1 G_1 + \dots + J_n G_n$ and $H_i \sim_m J_1 H_1 + \dots + J_n H_n$ 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 5 of Section 4 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 Fact 2 of Section 4, $G_j \sim_{m-|u|} (F_1 \cdot u)G_1 + \dots + (F_n \cdot u)G_n$ 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, $G_i \sim_m J_1 G_1 + \dots + J_n G_n$ and $H_i \sim_m J_1 H_1 + \dots + J_n H_n$ and no $J_i = \varepsilon$. Using Fact 2 of Section 4, $J_1 G_1 + \dots + J_n G_n \sim_{m-|u|} (F_1 \cdot u)G_1 + \dots + (F_n \cdot u)G_n$ and $J_1 H_1 + \dots + J_n H_n \not\sim_{m-|u|} (F_1 \cdot u)H_1 + \dots + (F_n \cdot u)H_n$. Now Proposition 5 of Section 4 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 $E_1 G_1^i + \dots + E_n G_n^i \doteq F_1 G_1^i + \dots + F_n G_n^i$ and each goal $h(i)$ has the form $E_1 H_1^i + \dots + E_n H_n^i \doteq F_1 H_1^i + \dots + F_n H_n^i$, 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 5 of Section 4 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, using Proposition 6 of Section 4, 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

7 Correctness of the decision procedure

In this section Theorems 2 and 3 of Section 5 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 $E \not\sim_n F$ by Fact 2.2 of Section 4. The proof rules are sound by Fact 1.2 and Proposition 2.2 of Section 5. 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, see Fact 1.1 and Proposition 2.1 of Section 5, 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 $E \doteq F$.*

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 $G_{i'} \doteq F'$ and $g(j)$ is $E' \doteq H_{j'}$ 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 $(F' \cdot w(X_j)) \doteq F''$.*
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 $E \doteq F$ 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 $E_i \doteq G_{i'}^1$ for some i' , otherwise BAL(L) can apply, and a goal $F_j \doteq H_{j'}^1$ 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 $g(0)$, $E \doteq F$, 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 $g(0)$, $E \doteq F$, 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 $(F' \cdot w(X_j)) \doteq F_i$ 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 $(F' \cdot w(X_j)) \doteq F_i$ 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 $g(i)$, $E'_i \doteq F'_i$ and $|F'_i| < |F'|$, and $g(j)$, $E'_j \doteq F'_j$ 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 $E \doteq F$ 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 $E \doteq F$ 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 $E' \doteq F'$ 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 $g(i)$, $E_i \doteq F_i$ where $|E_i| = 1$, and $g(j)$, $E_j \doteq F_j$ 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 $E_{i''} \doteq F_{i''}$ with $|E_{i''}| = 1$ and $E_{j''} \doteq F_{j''}$ with $|F_{j''}| = 1$: otherwise, BAL can apply. Therefore, every goal $g(j')$, $E_{j'} \doteq F_{j'}$, $j' \geq j$, 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 5, 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 $F_i = (F_0 \cdot u_1 \dots u_i)$ (by Corollary 1 of Section 5). 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 ε 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 $E_1 G_1 + \dots + E_n G_n \doteq F_1 G_1 + \dots + F_n G_n$, 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 $E_1(F_{j+1} \cdot w(X_1)) + \dots + E_k(F_{j+1} \cdot w(X_k)) \doteq F'$. 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 $g(i)$, $E_1^i G_1^i + \dots + E_{n_i}^i G_{n_i}^i \doteq F_1^i G_1^i + \dots + F_{n_i}^i G_{n_i}^i$, $i : 0 \leq i \leq t$ 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 by Proposition 6 of Section 4 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, $g'(i)$, $E_1 G_1^i + \dots + E_n G_n^i \doteq F_1 G_1^i + \dots + F_n G_n^i$, $1 \leq i \leq 2$, and $h'(i)$, $E_1 H_1^i + \dots + E_n H_n^i \doteq F_1 H_1^i + \dots + F_n H_n^i$, $1 \leq i \leq 2$, 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 5 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 $E_1^1 G_1^1 + \dots + E_{n_1}^1 G_{n_1}^1 \doteq F_1^1 G_1^1 + \dots + F_{n_1}^1 G_{n_1}^1$ and the application of BAL using F_{i_2} , the goal $g(k_2)$, has head/tail form $E_1^2 G_1^2 + \dots + E_{n_2}^2 G_{n_2}^2 \doteq F_1^2 G_1^2 + \dots + F_{n_2}^2 G_{n_2}^2$ 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

References

- [1] Burkart, O., Caucal, D., Moller, F., and Steffen, B. (2001). Verification on infinite structures. In *Handbook of Process Algebra*, edited Bergstra, J., Ponse, A., and Smolka, S. 545-623, *North Holland*.
- [2] 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.
- [3] Christensen, S., Hüttel, H., and Stirling, C. (1995). Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, **121**, 143-148.
- [4] Ginsberg, S., and Greibach, S. (1966). Deterministic context-free languages. *Information and Control*, 620-648.
- [5] Harrison, M. (1978). *Introduction to Formal Language Theory*, Addison-Wesley.
- [6] Harrison, M., and Havel, I. (1973). Strict deterministic grammars. *Journal of Computer and System Sciences*, **7**, 237-277.
- [7] Harrison, M., Havel, I., and Yehudai, A. (1979). On equivalence of grammars through transformation trees. *Theoretical Computer Science*, **9**, 173-205.

- [8] Hopcroft, J., and Ullman, J. (1979). *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley.
- [9] 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.
- [10] Korenjak, A and Hopcroft, J. (1966). Simple deterministic languages. *Procs. 7th Annual IEEE Symposium on Switching and Automata Theory*, 36-46.
- [11] Oyamaguchi, M., Honda, N., and Inagaki, Y. (1980). The equivalence problem for real-time strict deterministic languages. *Information and Control*, **45**, 90-115.
- [12] Sénizergues, G. (1997). The equivalence problem for deterministic push-down automata is decidable. *Lecture Notes in Computer Science*, **1256**, 671-681.
- [13] Sénizergues, G. (2001). $L(A) = L(B)$? decidability results from complete formal systems. *Theoretical Computer Science*, **251**, 1-166.
- [14] Sénizergues, G. (2001). $L(A) = L(B)$? a simplified decidability proof. pp.1-58. To appear in *Theoretical Computer Science*.
- [15] Stirling, C. (1998). Decidability of bisimulation equivalence for normed pushdown processes. *Theoretical Computer Science*, **195**, 113-131.
- [16] Stirling, C. (2001). Decidability of DPDA equivalence. *Theoretical Computer Science*, **255**, 1-31.