Division of Informatics, University of Edinburgh

Laboratory for Foundations of Computer Science

Decidability of Bisimulation Equivalence for Pushdown Processes

by

Colin Stirling

Informatics Research Report EDI-INF-RR-0005

Division of Informatics **January 2000**
http://www.informatics.ed.ac.uk/

Decidability of Bisimulation Equivalence for Pushdown Processes

Colin Stirling

Informatics Research Report EDI-INF-RR-0005

DIVISION *of* INFORMATICS
Laboratory for Foundations of Computer Science

January 2000

Abstract : We show that bisimulation equivalence is decidable for pushdown automata without epsilon-transitions.

Keywords : Decidability,Bisimulation Equivelance,Pushdown Automata

Decidability of Bisimulation Equivalence for Pushdown Processes

Colin Stirling
Division of Informatics
University of Edinburgh
Edinburgh EH9 3JZ, UK

email: cps@dcs.ed.ac.uk

Abstract

We show that bisimulation equivalence is decidable for pushdown automata without ϵ-transitions.

1 Introduction

Automata and language theory studies finitely presented mechanisms for generating families of words. The Chomsky hierarchy of languages can be generated using grammars or using automata. For example pushdown automata were introduced as a counterpart to context-free grammars. Both accept the same languages, the context-free languages. A slight shift in focus is very revealing. Instead of grammars and automata as language generators, one views them as propagators of labelled transition graphs. Pushdown automata are then strictly more expressive than context-free grammars. Concurrency theory as developed using process calculi views the behaviour of a process as a transition graph. Consequently we can view configurations of grammars and automata as processes. Concurrency theory requires a more intensional exposition of behaviour than provided by language theory because language equivalence is not preserved by communicating automata. Numerous finer equivalences have been proposed. Bisimulation equivalence, due to Park and Milner, has received much attention.

In this paper we show that bisimulation equivalence is decidable for pushdown processes (which are pushdown automata without ϵ-transitions). The result follows a line of research initiated by Baeten, Bergstra and Klop [1] who proved that bisimulation equivalence is decidable for normed context-free processes (which

was then generalised to all context-free processes in [7]). The author showed decidability of bisimilarity for normed pushdown processes in [17]. It is this result which is generalised here.

An important development in the technology for showing decidability was the proof of decidability of language equivalence for deterministic pushdown automata, DPDA, by Sénizergues [13, 14]. DPDA include ϵ-transitions which are not considered here. This problem had been open for over thirty years. However the proof is formidable, and when spelt out in full takes up over 70 pages of the full 166 page paper [14]. Sénizergues generalised this proof to decidability of bisimilarity for PDA with deterministic popping ϵ-transitions [15]. Although a generalisation of the case considered here, the proof is even more formidable than his DPDA proof (and is over 100 pages long [16]). A simplification of the DPDA equivalence proof was presented by the author [18] using ideas from concurrency theory (for showing decidability of bisimilarity [10, 17]) and insights from Sénizergues's intricate proof. Using techniques within this proof, and a notion of determinization, we provide here a manageable proof of decidability of bisimulation equivalence for pushdown processes: in fact the method extends to the more general case considered by Sénizergues, as we note later.

2 Pushdown processes

Ingredients for pushdown automata without ϵ-transitions, PDA, over a finite alphabet A are a finite set of states $\mathsf{P} = \{p_1, \ldots, p_k\}$, a finite set of stack symbols $\Gamma = \{X_1, \ldots, X_n\}$ and a finite set of basic transitions, each of the form $pX \stackrel{a}{\longrightarrow} q\alpha$ where p and q are states in P, $a \in \mathsf{A}$, X is a stack symbol and α is a sequence of stack symbols. A pushdown configuration or process is any expression of the form $p\alpha$ where $p \in \mathsf{P}$ and $\alpha \in \Gamma^*$, whose transition graph is determined by the basic transitions together with the following prefix rule, where $\beta \in \Gamma^*$

$$\text{if } pX \stackrel{a}{\longrightarrow} q\alpha \text{ then } pX\beta \stackrel{a}{\longrightarrow} q\alpha\beta$$

This account follows the presentation of Caucal [4]. It is a slight redescription of classical PDA without ϵ-transitions and final states, as for instance in [9] viewing them as transition graph generators instead of language acceptors. The extended transition relation $p\alpha \stackrel{w}{\longrightarrow} q\beta$ for words $w \in \mathsf{A}^*$ is defined in the usual way.

No restictions are imposed on a PDA. In [17] we assumed normedness, which is the condition that for every configuration $p\alpha$ there is word w such that $p\alpha \stackrel{w}{\longrightarrow} q\epsilon$ for some state q. In particular a process may only have perpetual behaviour, or it may be a "sink" without transitions. In fact we can "normalise" the PDA so that it avoids sink configurations. First one can assume that for any basic transition $pX \stackrel{a}{\longrightarrow} q\alpha$ the length of α, written $|\alpha|$, is at most 2. Auxiliary stack symbols can be introduced to ensure this condition. Next one can assume that the PDA

contains no sinks. Let t be a new letter not in the alphabet. For any state p and any stack symbol X such that pX has no transitions add a new transition $pX \stackrel{t}{\longrightarrow} pX$, so t registers termination. Next assume a new stack symbol Y such that for every state p there is a basic transition $pY \stackrel{t}{\longrightarrow} pY$. Two configurations $p\alpha$ and $q\beta$ are bisimulation equivalent in the original PDA with sinks iff $p\alpha Y$ and $q\beta Y$ are bisimulation equivalent in the normalised PDA[1]. In the following we assume normalised PDA.

Example 1 $\mathsf{P} = \{p, q, r, s\}$, $\Gamma = \{X, Z\}$ and $\mathsf{A} = \{a, b, c, d\}$. The basic transitions are as follows.

$$\begin{array}{llll} pZ \stackrel{a}{\longrightarrow} pXZ & qZ \stackrel{d}{\longrightarrow} pZ & pX \stackrel{a}{\longrightarrow} pXX & qX \stackrel{c}{\longrightarrow} q\epsilon \\ pZ \stackrel{b}{\longrightarrow} qZ & rZ \stackrel{d}{\longrightarrow} pZ & pX \stackrel{b}{\longrightarrow} qX & rX \stackrel{c}{\longrightarrow} s\epsilon \\ pZ \stackrel{b}{\longrightarrow} rZ & sZ \stackrel{c}{\longrightarrow} rZ & pX \stackrel{b}{\longrightarrow} rX & sX \stackrel{c}{\longrightarrow} rX \end{array}$$

The transition graph generated by pZ is as follows.

$$\begin{array}{ccccccccc} rZ & \stackrel{c}{\longleftarrow} & sZ & \stackrel{c}{\longrightarrow} & rXZ & \stackrel{c}{\longleftarrow} & sXZ & \stackrel{c}{\longleftarrow} & rXXZ \stackrel{c}{\longleftarrow} \ldots \\ d\downarrow\uparrow b & & & & \uparrow b & & & & \uparrow b \\ pZ & \stackrel{a}{\longrightarrow} & & & pXZ & \stackrel{a}{\longrightarrow} & & & pXXZ \stackrel{a}{\longrightarrow} \ldots \\ d\uparrow\downarrow b & & & & \downarrow b & & & & \downarrow b \\ qZ & \stackrel{c}{\longleftarrow} & & & qXZ & \stackrel{c}{\longleftarrow} & & & qXXZ \stackrel{c}{\longleftarrow} \ldots \end{array}$$

For any $n > 0$ the transition $pX^nZ \stackrel{b}{\longrightarrow} qX^nZ$ is derived from the basic transition $pX \stackrel{b}{\longrightarrow} qX$ using the prefix rule when β is $X^{n-1}Z$. □

Example 1 illustrates how a pushdown configuration may generate an infinite state transition graph. Such a graph exhibits a "regular" structure, as defined by Muller and Schupp [12]. A rooted connected graph of bounded degree over a finite alphabet is said to be "context-free" if it contains only finitely many isomorphism classes under end-isomorphisms. Let G be a labelled transition graph of bounded in and out degree with root v. A vertex u has distance n from the root v if there is a shortest path from v to u whose length is n, where this path can traverse transitions in both directions. In Example 1 sZ has distance 2 from pZ. Let G_n be the subgraph of G when all vertices whose distance from v is less than n are removed. If u has distance n from the root then let $G_n(u)$ be the connected subgraph of G_n rooted at u. G is context-free if the family

$$\bigcup_{n \geq 0} \{G_n(u) : u \text{ has distance } n \text{ from the root } v \text{ in } G\}$$

has only finitely many isomorphism classes. Muller and Schupp show that the transition graphs of pushdown processes are exactly the "context-free" graphs.

[1]The definition of bisimulation equivalence appears below in Definition 1.

This characterisation of pushdown processes is independent of syntactic mechanisms, such as grammars and automata, for generating graphs.

Caucal provided two further characterisations of the graphs of pushdown processes. First he showed that they can be defined in terms of "pattern graphs" using deterministic graph grammars [4]. Secondly they are the transition graphs generated by Type 0 grammars under prefix rewriting. Assume a finite family of nonterminals N and a finite alphabet A. A Type 0 grammar under prefix rewriting is given by a finite family of basic transitions of the form $\alpha \stackrel{a}{\longrightarrow} \beta$ where α and β belong to N^*. The transition graph generated by a configuration $\delta \in \mathsf{N}^*$ is determined by the basic transitions together with the prefix rule if $\alpha \stackrel{a}{\longrightarrow} \beta$ then $\alpha\gamma \stackrel{a}{\longrightarrow} \beta\gamma$ for any $\gamma \in \mathsf{N}^*$.

In concurrency theory the behaviour of a process is represented as a labelled transition graph. A more intensional account of process equivalence is needed than that given by recognising the same language (as language equivalence is not preserved by communicating automata). A pivotal notion, due to Park and Milner, is bisimilarity which is finer than language equivalence on processes.

Definition 1 A binary relation R between processes is a bisimulation relation provided that whenever $(E, F) \in R$, for all labels a,

$$\begin{array}{l} \text{if } E \stackrel{a}{\longrightarrow} E' \text{ then } \exists F'.F \stackrel{a}{\longrightarrow} F' \text{ and } (E', F') \in R, \\ \text{if } F \stackrel{a}{\longrightarrow} F' \text{ then } \exists E'.E \stackrel{a}{\longrightarrow} E' \text{ and } (E', F') \in R. \end{array}$$

Two processes E and F are bisimulation equivalent, or bisimilar, written $E \sim F$, if there is a bisimulation relation R relating them. For instance, $rXZ \sim qXXZ$ but $rXZ \not\sim qXZ$ in Example 1.

Caucal [4] showed that with respect to bisimilarity pushdown automata are more expressive than context-free grammars. Example 1 is an illustrative instance: there is not a context-free grammar with root configuration C such that $C \sim pZ$. Burkart and Steffen provide additional insight [3] by showing that, unlike context-free grammars, pushdown automata are closed under Hoare parallel composition with finite-state processes (with respect to bisimilarity). Moreover they demonstrate that the family of pushdown automata is the smallest extension of context-free grammars with this closure property.

Baeten et al. proved that bisimulation equivalence is decidable for normed context-free grammars [1]. The decidability result was generalised in [7] to all context-free grammars, and then refined in [2] to give an exponential upper bound. Groote and Hüttel proved that other standard equivalences on processes (traces, failures, simulation, $\frac{2}{3}$-bisimulation, etc.) are all undecidable for context-free grammars [8] (and therefore for pushdown automata). The author showed decidability of bisimulation equivalence for normed pushdown automata [17]. In the special case that the stack symbols Γ contains just one element, decidability of bisimilarity for pushdown automata without the restriction to normedness was

shown by Jancar [11]. The key idea, elegantly presented using regular colourings, in this restricted case is that a witness for bisimilarity is semi-linear.

An important development in the technology for proving decidability was the proof of decidability of language equivalence for deterministic pushdown automata, DPDA, by Sénizergues [13, 14]. He generalised this proof to decidability of bisimilarity for PDA with deterministic popping ϵ-transitions [15, 16]. Although more general than the case considered here, the proof is very intricate and very long (over 100 pages). The author simplified Sénizergues's proof of decidability of language equivalence for DPDA [18]. This proof uses ideas from concurrency theory (in particular tableaux and decomposition for proving decidability of bisimilarity [10, 17]) as well as insights from Sënizergues's proof. It is our intention here to show that there is a manageable proof of decidability of bisimulation equivalence for pushdown processes using a similar technique together with a means for determinizing. In fact the method also generalises to the case considered by Sénizergues with ϵ-transitions, as we note in the conclusion.

Decidability of bisimilarity for pushdown processes is harder to show than for context-free grammars. The structural method used in [7], which appeals to decomposition and congruence, is not immediately applicable because it is not clear what are the components of a pushdown configuration. A configuration of a context-free grammar in Greibach normal form is just a sequence of nonterminals $N_1 \ldots N_m$ and is therefore built from the components N_i, $1 \leq i \leq m$. A pushdown process $pX_1 \ldots X_m$ does not contain X_i as a component. Part of the decidability proof is to expose enough structure within pushdown processes to permit decomposition into components.

The proof of decidability consists of two semi-decision procedures (for which we are unable to provide a complexity measure, as is the case with Sénizergues's proof). One half of the proof is straightforward because bisimilarity is characterizable using approximants when processes, such as pushdown processes, are image-finite: E is image-finite if for each $w \in \mathsf{A}^*$, the set $\{F : E \stackrel{w}{\longrightarrow} F\}$ is finite.

Definition 2 The family $\{\sim_n : n \geq 0\}$ is defined inductively as follows:

$$
\begin{array}{l}
E \sim_0 F \text{ for all } E, F \\
E \sim_{n+1} F \text{ iff for all } a \in \mathsf{A} \\
\quad \text{if } E \stackrel{a}{\longrightarrow} E' \text{ then } \exists F'.F \stackrel{a}{\longrightarrow} F' \text{ and } E' \sim_n F', \text{ and} \\
\quad \text{if } F \stackrel{a}{\longrightarrow} F' \text{ then } \exists E'.E \stackrel{a}{\longrightarrow} E' \text{ and } E' \sim_n F'
\end{array}
$$

The following is a standard result.

Proposition 1 *If E and F are image-finite then $E \sim F$ iff $\forall n \geq 0.\ E \sim_n F$.*

For each $n \geq 0$, the relation $\sim_n$ on pushdown processes is decidable, and therefore bisimulation inequivalence is semi-decidable using the simple procedure which

seeks the least i such that $p\alpha \not\sim_i q\beta$. Therefore we just need to establish semi-decidability of bisimilarity. The crux of this part of the proof is that there is a finite tableau proof of $p\alpha \sim q\beta$. As finite proofs can be enumerated, this amounts to a semi-decision procedure.

In the next section we expose structure within pushdown configurations using auxiliary stack symbols, and in section 4 we introduce the important notion of a bisimulation witness. In sections 5 and 6 we isolate crucial combinatorial properties which underpin the tableau proof system which is presented in section 7.

3 Measures and auxiliary stack symbols

Assume a fixed normalised PDA over a finite alphabet A with state set $\mathsf{P} = \{p_1, \ldots, p_k\}$ and stack symbols Γ. The decision problem is "are two processes $p\alpha$ and $q\beta$ (where p, q belong to P and α and β are in Γ^+) bisimilar?"[2]. We shall now introduce a variety of supplementary notions that will be used in the decidability proof.

A total ordering is assumed on the state set P, so that $p_1 < \ldots < p_k$. The size of a process $p\alpha$ is is the length of α, $|\alpha|$. For each stack symbol X and state p the norm of pX, written $\mathrm{n}(pX)$, is a k-tuple $(w_1, \ldots, w_k)$ where each $w_i \in \mathsf{A}^+ \cup \{\bot\}$. The component w_i is either a shortest length word such that $pX \stackrel{w_i}{\longrightarrow} p_i\epsilon$ or there is no such word and $w_i = \bot$. The notation $\mathrm{n}(pX)_i$ stands for the ith element of $\mathrm{n}(pX)$. Note that norm is easily computable. We say that pX is unnormed if $\mathrm{n}(pX)_i$ is $\bot$ for all i. If pX is unnormed then it has only perpetual behaviour, and therefore it follows that $pX \sim pX\alpha$ for any α. An important measure of the PDA is M which is just larger than the size of the longest word in a norm:

$$\mathrm{M} \stackrel{\mathrm{def}}{=} 1 + \max\{\, |w_i| \in \mathsf{A}^+ : w_i \text{ is an entry in some } \mathrm{n}(pX)\}$$

Assume a process $p\alpha\delta$. We need a notion of its "components". By itself δ is not a process. However for each state p_i, $p_i\delta$ can be viewed as a "part" of $p\alpha\delta$. Therefore one can think of $p\alpha\delta$ as implicitly having the form $p\alpha(p_1\delta, \ldots, p_k\delta)$ where the bracketing notation $(p_1\delta, \ldots, p_k\delta)$ picks out the continuing behaviour depending on the terminating state of $p\alpha$: if $p\alpha \stackrel{u}{\longrightarrow} p_j\epsilon$ then $p\alpha(p_1\delta, \ldots, p_k\delta) \stackrel{u}{\longrightarrow} p_j\delta$. This implicit form allows substitutivity of equivalents: if $p_i\delta \sim r_i\gamma_i$ for each i then it follows that $p\alpha\delta \sim p\alpha(r_1\gamma_1, \ldots, r_k\gamma_k)$. This can be slightly refined because not every component $p_i\delta$ may be accessible. Let $\mathrm{T}(p\alpha)$ be the indices of the "terminating" state set, $\{i : p\alpha \stackrel{w}{\longrightarrow} p_i\epsilon \text{ for some } w\}$. If $p_i\delta \sim r_i\gamma_i$ for each $i \in \mathrm{T}(p\alpha)$ then $p\alpha\delta \sim p\alpha(r_1\gamma_1, \ldots, r_k\gamma_k)$. However instead of explicitly introducing bracketing it is introduced implicitly using auxiliary stack symbols.

[2]Because the disjoint union of two PDAs is a PDA, one can assume the decision question over configurations of a single PDA instead of between two different PDAs.

The fixed PDA is extended with a finite family of auxiliary stack symbols W. Each new stack symbol W has an associated definition $W \stackrel{\text{def}}{=} (q_1\delta_1, \ldots, q_k\delta_k)$ where each $q_i \in \mathsf{P}$ and each δ_i is a sequence of stack elements, possibly including auxiliary symbols. Let W_i be the ith component of W, the configuration $q_i\delta_i$. For each state $p_j \in \mathsf{P}$ the behaviour of p_jW is that of $q_j\delta_j$, the jth component of W. An extended pushdown process is an expression of the form $p\alpha$ where $\alpha \in (\Gamma \cup \mathsf{W})^*$. Basic transitions still have the form $pX \stackrel{a}{\longrightarrow} q\alpha$, where $X \in \Gamma$ and $\alpha \in \Gamma^*$. However the prefix rule is slightly generalised as follows (which turns out to be sufficient for the processes considered later).

$$\begin{array}{l}
\text{if } pX \stackrel{a}{\longrightarrow} q\alpha \text{ and } \alpha \neq \epsilon \text{ then } pX\beta \stackrel{a}{\longrightarrow} q\alpha\beta \\
\text{if } pX \stackrel{a}{\longrightarrow} p_i\epsilon \text{ and } Y \in \Gamma \text{ then } pXY\beta \stackrel{a}{\longrightarrow} p_iY\beta \\
\text{if } pX \stackrel{a}{\longrightarrow} p_i\epsilon \text{ and } W \stackrel{\text{def}}{=} (q_1\delta_1, \ldots, q_k\delta_k) \text{ then } pXW\beta \stackrel{a}{\longrightarrow} q_i\delta_i\beta
\end{array}$$

The auxiliary family of stack symbols W is partitioned into two. First are *simple* elements, each of which has a definition $U \stackrel{\text{def}}{=} (q_1\gamma_1, \ldots, q_k\gamma_k)$ where each $\gamma_i \in \Gamma^*$ and $0 \leq |\gamma_i| \leq 2\text{M}$. Auxiliary stack symbols are not allowed in their definition. Therefore there are only finitely many different simple stack elements. The other elements in W are *recursive*. A recursive stack symbol can only appear as a final element in a process. Each recursive stack symbol has a definition $V \stackrel{\text{def}}{=} (q_1\lambda_1V, \ldots, q_k\lambda_kV)$ where λ_i is a sequence of stack symbols which may contain simple but not recursive auxiliary stack symbols, and V is the defining element. A component V_i for which λ_i is ϵ is said to be a "terminating configuration". We impose some conditions on these configurations. If $V_i = p_jV$ then $p_j \leq p_i$ (where $<$ is the ordering on P), and $V_j = p_jV$. Therefore by the transition rules terminating configurations are "sinks"[3]. Although there is no upper bound on the number of different recursive stack symbols, in the decidability proof we only appeal to a finite subset of them.

In the decidability proof we are only interested in particular pairs of processes $p\alpha$ and $q\beta$. First is the case when neither contain recursive auxiliary symbols. Second is the case that both contain the same recursive stack symbol, and so α then has the form $\alpha'V$ and β has the form $\beta'V$. We never need to consider the case where the pair of processes contains different recursive stack symbols. Next is a slight refinement of the definition of bisimulation equivalence to take account of terminating configurations.

Definition 1 A binary relation R between extended PDA processes is a bisimulation relation provided that whenever $(p\alpha, q\beta) \in R$, for all $a \in \mathsf{A}$,

$$\begin{array}{l}
\text{if } V_i = p_iV \text{ then } p = p_i \text{ and } \alpha = V \text{ iff } q = p_i \text{ and } \beta = V, \text{ and} \\
\text{if } p\alpha \stackrel{a}{\longrightarrow} p'\alpha' \text{ then } \exists q'\beta'.q\beta \stackrel{a}{\longrightarrow} q'\beta' \text{ and } (p'\alpha', q'\beta') \in R, \text{ and} \\
\text{if } q\beta \stackrel{a}{\longrightarrow} q'\beta' \text{ then } \exists p'\alpha'.p\alpha \stackrel{a}{\longrightarrow} p'\alpha' \text{ and } (p'\alpha', q'\beta') \in R.
\end{array}$$

[3]Note that sinks have been introduced into extended PDA processes: the reason for this will become clearer below, and as the proof proceeds.

As before two processes are bisimilar if there is a bisimulation relation R relating them. The relation $\sim$ is still used for the equivalence. The equivalence between two processes which do not contain recursive stack symbols is not affected by this refinement. However two processes with recursive symbols must agree on their terminating states.

The definition of bisimulation approximant is similarly refined.

Definition 2 The family $\{\sim_n : n \geq 0\}$ on extended pushdown processes is defined inductively as follows:

$$
\begin{array}{l}
pX\alpha \sim_0 qY\beta \text{ if } X, Y \in \Gamma, \text{ and} \\
\text{if } V_i = p_i V \text{ then } p_i V \sim_0 p_j V \text{ iff } p_i = p_j \\
p\alpha \sim_{n+1} q\beta \text{ iff } p\alpha \sim_0 q\beta \text{ and for all } a \in \mathsf{A} \\
\quad \text{if } p\alpha \xrightarrow{a} p'\alpha' \text{ then } \exists q'\beta'.q\beta \xrightarrow{a} q'\beta' \text{ and } p'\alpha' \sim_n q'\beta', \text{ and} \\
\quad \text{if } q\beta \xrightarrow{a} q'\beta' \text{ then } \exists p'\alpha'.p\alpha \xrightarrow{a} p'\alpha' \text{ and } p'\alpha' \sim_n q'\beta'
\end{array}
$$

The following result follows from image-finiteness of pushdown processes.

Fact 1 *$p\alpha \sim q\beta$ iff $\forall n \geq 0$. $p\alpha \sim_n q\beta$.*

The new definition of bisimilarity precludes terminating configurations pV and qV being equivalent except when state p is the same as q. One reason for this slightly more intensional notion of bisimilarity is to support refinement of recursive stack symbols. A recursive stack symbol $V \stackrel{\text{def}}{=} (q_1\lambda_1 V, \ldots, q_k\lambda_k V)$ is said to "refine" another stack symbol $W \stackrel{\text{def}}{=} (r_1\gamma_1 W, \ldots, r_k\gamma_k W)$ if the following two conditions hold

$$
\begin{array}{l}
\text{if } \gamma_i \neq \epsilon \text{ then } q_i = r_i \text{ and } \lambda_i = \gamma_i \\
\text{if } W_i = W_j \text{ then } V_i = V_j
\end{array}
$$

A refined recursive stack symbol agrees on the definitions of nonterminating components and preserves equality of definitions, but may contain fewer terminating components. Because equivalence of processes includes agreement of state in the case of terminating configurations, equivalence is preserved by refinement.

Fact 2 *If V refines W and $p\alpha W \sim q\beta W$ then $p\alpha V \sim q\beta V$.*

4 Bisimulation witnesses

This section is devoted to another auxiliary notion that is used in the decidability proof. A "bisimulation witness" D for a pair of nonidentical processes $(p\alpha, q\beta)$, where $p \neq q$ or $\alpha \neq \beta$, is a tree of depth $d \geq 0$ which is a partial witness for showing that $p\alpha$ and $q\beta$ are bisimilar. The root of D is $(p\alpha, q\beta)$. If D has depth 0 then it consists of the single root node $(p\alpha, q\beta)$. Otherwise for each transition $p\alpha \xrightarrow{a_i} p_i\alpha_i$ there is a transition $q\beta \xrightarrow{a_i} q_i\beta_i$ and for each transition

$q\beta \xrightarrow{a_i} q_i\beta_i$ there is a transition $p\alpha \xrightarrow{a_i} q_i\alpha_i$ such that there is a subtree D' for $(p_i\alpha_i, q_i\beta_i)$ of depth $d-1$ directly beneath the root node $(p\alpha, q\beta)$. We write $D \xrightarrow{a_i} D'(p_i\alpha_i, q_i\beta_i)$, or more succinctly $D \xrightarrow{a_i} D_i$ when the new root is known, to represent this situation. A further condition required of a witness D is that if $(p'\alpha', q'\beta')$ is a leaf node of D then $p'\alpha' \sim_0 q'\beta'$. If a pair of processes $(p\alpha, q\beta)$ are equal, $p = q$ and $\alpha = \beta$, then we assume that its witness is the special identity witness I consisting of the single node $(p\alpha, q\beta)$. I counts has having arbitrary depth.

The depth of a witness D is denoted by $\mathrm{d}(D)$. If D is a witness for $(p\alpha, q\beta)$ with depth d then it follows immediately that $p\alpha \sim_d q\beta$. Because $p\alpha \sim_n p\alpha$ for any n it is not necessary to unravel $I(p\alpha, p\alpha)$ into a tree of depth n. The transition relation for subwitnesses is extended to words. If $w = a_1 \ldots a_n$ and $D \xrightarrow{a_1} \ldots \xrightarrow{a_n} D_n$ then this is abbreviated to $D \xrightarrow{w} D_n$.

The definition of bisimilarity is extended to cover when $p\alpha$ is bisimilar to $q\beta$ relative to witness $D(p\alpha, q\beta)$, written $D \models p\alpha \sim q\beta$.

Definition 1 $D \models p\alpha \sim q\beta$ is defined inductively on $\mathrm{d}(D)$ as follows.

1. If $d(D) = 0$ then $D \models p\alpha \sim q\beta$ iff $p\alpha \sim q\beta$.
2. If $d(D) > 0$ and $D \neq I$ then $D \models p\alpha \sim q\beta$ iff $D' \models p_i\alpha_i \sim q_i\beta_i$ for all witnesses $D'(p_i\alpha_i, q_i\beta_i)$ such that $D \xrightarrow{a_i} D'$ for all $a_i \in \mathsf{A}$.
3. $I \models p\alpha \sim p\alpha$.

The definition of bisimulation approximant is also extended to when $p\alpha \sim_n q\beta$ relative to witness $D(p\alpha, q\beta)$, written $D \models p\alpha \sim_n q\beta$.

Definition 2 $D \models p\alpha \sim_n q\beta$ is defined by induction on n and $\mathrm{d}(D)$ as follows.

1. If $n = 0$ then $D \models p\alpha \sim_n q\beta$.
2. If $d(D) = 0$ then $D \models p\alpha \sim_n q\beta$ iff $p\alpha \sim_n q\beta$.
3. If $d(D) > 0$ and $n > 0$ and $D \neq I$ then $D \models p\alpha \sim_n q\beta$ iff $D' \models p_i\alpha_i \sim_{n-1} q_i\beta_i$ for all witnesses $D'(p_i\alpha_i, q_i\beta_i)$ such that $D \xrightarrow{a_i} D'$ for all $a_i \in \mathsf{A}$.
4. $I \models p\alpha \sim_n p\alpha$.

The following results are straightforward to show.

Fact 1

1. *If $p\alpha \sim q\beta$ then for all $m \geq 0$ there is a witness D for $(p\alpha, q\beta)$ of depth m such that $D \models p\alpha \sim q\beta$.*
2. *If $p\alpha \sim_n q\beta$ then for all $m : 0 \leq m \leq n$ there is a witness D for $(p\alpha, q\beta)$ of depth m such that $D \models p\alpha \sim_n q\beta$.*

3. *For any pair* $(p\alpha, q\beta)$ *and depth* d *there are only finitely many different witnesses* $D(p\alpha, q\beta)$ *with* $\mathrm{d}(D) \leq d$.

The decision procedure for bisimilarity utilises tree surgery on witnesses. Here we consider some useful basic operations on, and relations between, witnesses. A witness D may be extended to a deeper witness D', written $D < D'$, by extending the leaves of D. Formally $D' > D$ if $\mathrm{d}(D') > \mathrm{d}(D)$ and D' restricted to depth $\mathrm{d}(D)$ is the witness D. If $D(p\alpha, q\beta)$ is a witness to depth d then D^c is the "converse" witness of depth d with root $(q\beta, p\alpha)$ which is the result of replacing all nodes $(p'\alpha', q'\beta')$ in D with $(q'\beta', p'\alpha')$. Witnesses may also be composed. If the root of D is $(p\alpha, q\beta)$ and the root of D' is $(q\beta, r\gamma)$ then the witness $D \circ D'$ has root $(p\alpha, r\gamma)$. If $\mathrm{d}(D)$ or $\mathrm{d}(D')$ equals 0 then $\mathrm{d}(D \circ D') = 0$. Otherwise if $D \stackrel{a}{\longrightarrow} D_1(p'\alpha', q'\beta')$ and $D' \stackrel{a}{\longrightarrow} D'_1(q'\beta', r'\gamma')$ then $D \circ D' \stackrel{a}{\longrightarrow} (D_1 \circ D'_1)(p'\alpha', r'\gamma')$. In the special case that D or D' is I, we assume that $D \circ I = D$ and $I \circ D' = D'$.

The following result captures elementary properties of witnesses which will be used in later sections.

Fact 2

1. *If* $D \models p\alpha \sim q\beta$ and $m > \mathrm{d}(D)$ *then there is a* $D' > D$ *with* $\mathrm{d}(D') = m$ *such that* $D' \models p\alpha \sim q\beta$.
2. *If* $D \models p\alpha \sim_n q\beta$ *and* $D' < D$ *then* $D' \models p\alpha \sim_n q\beta$.
3. $I \models p\alpha \sim p\alpha$.
4. $D \models p\alpha \sim q\beta$ *iff* $D^c \models q\beta \sim p\alpha$.
5. $D \models p\alpha \sim_n q\beta$ *iff* $D^c \models q\beta \sim_n p\alpha$.
6. *If* $D \models p\alpha \sim q\beta$ *and* $D' \models q\beta \sim r\gamma$ *then* $D \circ D' \models p\alpha \sim r\gamma$.
7. *If* $D \models p\alpha \sim_n q\beta$ *and* $D' \models q\beta \sim_n r\gamma$ *then* $D \circ D' \models p\alpha \sim_n r\gamma$.
8. *If* $\mathrm{d}(D) \geq \mathrm{d}(D')$ *and* $D \models p\alpha \sim_n q\beta$ *and* $D' \models q\beta \not\sim_n r\gamma$ *then* $D \circ D' \models p\alpha \not\sim_n r\gamma$.
9. *If* $\mathrm{d}(D') \geq \mathrm{d}(D)$ *and* $D \models p\alpha \not\sim_n q\beta$ *and* $D' \models q\beta \sim_n r\gamma$ *then* $D \circ D' \models p\alpha \not\sim_n r\gamma$.

5 Imbalance

Consider trying to show that $p\alpha \sim q\beta$. One approach is goal directed. Start with the goal $p\alpha = q\beta$ (to be understood as "is it true that $p\alpha \sim q\beta$?") and then reduce it to subgoals. Keep reducing to further subgoals until one reaches either

obviously true subgoals, such as $r\delta = r\delta$, or obviously false subgoals, such as $r\lambda = s\gamma$ where $r\lambda \not\sim_1 s\gamma$. This naive technique which is described more formally later in terms of tableaux is the approach adopted, except that goals will also involve bisimulation witnesses.

The "imbalance" between a pair of processes is the length of the longest prefix of stack sequences before they have a common tail. For instance, the imbalance between $p\alpha\delta$ and $q\beta\delta$ is $\max\{|\alpha|, |\beta|\}$. An important step in the proof is that imbalance can be bounded when reducing goals to subgoals. The following result utilises the feature of a normalised PDA that if γ does not contain auxiliary stack symbols and $p\gamma \stackrel{w}{\longrightarrow} p'\gamma'$ then $|\gamma| - |w| \leq |\gamma'| \leq |\gamma| + |w|$.

Proposition 1 *If $pX\alpha \sim q\beta\delta$ and $|\beta| = \mathrm{M}+1$ and β does not contain auxiliary stack elements and $|u| = \mathrm{M}$ and $pX \stackrel{u}{\longrightarrow} p'\alpha'$ and $\alpha' \neq \epsilon$ and $q\beta \stackrel{u}{\longrightarrow} q'\beta'$ and $p'\alpha'\alpha \sim q'\beta'\delta$ then there is a simple stack symbol U such that $p'\alpha'U\delta \sim q'\beta'\delta$.*

Proof: Assume $pX\alpha \sim q\beta\delta$ and $|\beta| = \mathrm{M}+1$, and β does not contain auxiliary stack elements. We define a simple stack symbol U by cases on the entries of $\mathrm{n}(pX)$. If $\mathrm{n}(pX)_i = w_i$ then $pX \stackrel{w_i}{\longrightarrow} p_i\epsilon$ and so $pX\alpha \stackrel{w_i}{\longrightarrow} p_i\alpha$. Because $pX\alpha \sim q\beta\delta$ it follows that $q\beta\delta \stackrel{w_i}{\longrightarrow} r_i\lambda$ such that $p_i\alpha \sim r_i\lambda$, for some $r_i\lambda$. However $|w_i| < \mathrm{M}$ and therefore $r_i\lambda$ has the form $r_i\gamma_i\delta$ where $1 < |\gamma_i| \leq 2\mathrm{M}$ and γ_i does not contain auxiliary stack symbols . Therefore the ith component of U is $r_i\gamma_i$ and $p_i\alpha \sim U_i\delta$. The other case is that $\mathrm{n}(pX)_i = \perp$ and then the ith component of U is $p_i\epsilon$. Therefore U is a simple stack symbol. Assume that $pX \stackrel{u}{\longrightarrow} p'\alpha'$ and $\alpha' \neq \epsilon$ and $q\beta \stackrel{u}{\longrightarrow} q'\beta'$ and therefore $\beta' \neq \epsilon$ $p'\alpha'\alpha \sim q'\beta'\delta$. Because $\mathrm{T}(p'\alpha') \subseteq \mathrm{T}(pX)$ and $p_i\alpha \sim U_i\delta$ for each $i \in \mathrm{T}(pX)$ the result $p'\alpha'U\delta \sim q'\beta'\delta$ follows. □

Proposition 1 can be used to reduce imbalance. If $pX\alpha \sim q\beta\delta$ and $G = p'\alpha'\alpha \sim q'\beta'\delta$ obey the conditions in the Proposition then the result permits G to be reduced to a true subgoal $p'\alpha'U\delta \sim q'\beta'\delta$, whose imbalance is bounded because $2 \leq |\alpha'U| \leq \mathrm{M}+2$ and $1 \leq |\beta'| \leq 2\mathrm{M}+1$. In fact Proposition 1 is only a prelude. It presupposes that the goals are true, and we want to reduce imbalance when trying to establish that they are true. Moreover the result only tells us that there is a simple stack symbol with the correct property. If $\mathrm{n}(pX)_i = w_i$ and $pX\alpha \stackrel{w_i}{\longrightarrow} p_i\alpha$. then there may be many processes $r_i\gamma_i$ such that $q\beta \stackrel{w_i}{\longrightarrow} r_i\gamma_i$, and not all of them may have the property that $p_i\alpha \sim r_i\gamma_i\delta$. A solution to both problems is to use bisimulation witnesses.

$D \models p\alpha = q\beta$ represents that D is a bisimulation witness whose root is $(p\alpha, q\beta)$. This is the goal, "is it true that $D \models p\alpha \sim q\beta$?", which may or may not be true. However if $\mathrm{d}(D) = m$ then at least $D \models p\alpha \sim_m q\beta$. Assume $D \models pX\alpha = q\beta\delta$ and $\mathrm{d}(D) \geq \mathrm{M}$ and $|\beta| = \mathrm{M}+1$ and β does not contain auxiliary stack symbols. Assume that $pX \stackrel{u}{\longrightarrow} p'\alpha'$ where $|u| = \mathrm{M}$ and $\alpha' \neq \epsilon$, and $D \stackrel{u}{\longrightarrow} D'(p'\alpha'\alpha, q'\beta'\delta)$ and G is the goal $D' \models p'\alpha'\alpha = q'\beta'\delta$. We show that G can be reduced to a balanced subgoal. The information as to which simple stack symbol to use is already contained in D because its depth is at least M. If

$\mathrm{n}(pX)_i = w_i$ then there is a subwitness D_i such that $D \xrightarrow{w_i} D_i(p_i\alpha, r_i\gamma_i\delta)$. The ith component of U, namely $r_i\gamma_i$[4], can be scanned from D. The other case is when $\mathrm{n}(pX)_i = \perp$ and then U_i is $p_i\epsilon$: there cannot then be a subwitness of D with root $(p_i\alpha, r\lambda)$. However it is also necessary to know which witness justifies an entry of U. An "extended" simple stack symbol is a pair $(U, \overline{D}_i)$ where U is a simple stack symbol and $\overline{D}_i$ is the family of justifying witnesses $D_i(p_i\alpha, U_i\delta)$ for each $i \in \mathrm{T}(pX)$. We say that $(U, \overline{D}_i)$ is an extended simple stack symbol for the goal $D \models pX\alpha = q\beta\delta$, when it is derived from D as described.

Using $(U, \overline{D}_i)$ the goal $D' \models p'\alpha'\alpha = q'\beta'\delta$ can be reduced to a balanced subgoal $D'' \models p'\alpha' U\delta = q'\beta'\delta$. The issue is the witness D'', which is the result of surgery on D' using the witnesses $\overline{D}_i$. The construction is iterative with intermediate trees which are not proper witnesses. Assume $D'_0 = D'$ and D'_m is the current tree in the iteration. Consider any subtree $E(p_j\alpha_j\alpha, r\lambda)$ of D'_m which has maximal depth and has α as a tail in the left process. There are two cases.

1. $\alpha_j \neq \epsilon$. The root of E is changed to $(p_j\alpha_j U\delta, r\lambda)$. The result of this change to D'_m is D'_{m+1} and the iteration continues.

2. $\alpha_j = \epsilon$. The subwitness E is replaced with the witness $D^c_j \circ E$ whose root is $(U_j\delta, r\lambda)$. The result of this change to D'_m is D'_{m+1} and the iteration continues.

After a finite number of steps the iteration terminates with the tree D'_s, whose root is $(p'\alpha' U\delta, q'\beta'\delta)$, because D' has finite depth and at each stage $\mathrm{d}(D'_{j+1}) = \mathrm{d}(D'_j)$. D'_s is a proper witness for $(p'\alpha' U\delta, q'\beta'\delta)$. Let $\mathsf{B}(D')$ be the result, D'_s, of this iterative transformation on D' using the extended witness $(U, \overline{D}_i)$. This is the required witness D''.

Introducing balanced subgoals is both "complete" and "sound". The exact formulation of soundness below will become clearer when the tableau proof system is introduced in section 7.

Proposition 2 *Assume* $|\beta| = \mathrm{M} + 1$ *and* β *does not contain auxiliary stack symbols and* $\mathrm{d}(D) \geq \mathrm{M}$ *and* $(U, \overline{D}_i)$ *is an extended simple stack symbol for* $D \models pX\alpha = q\beta\delta$. *Also assume* $pX \xrightarrow{u} p'\alpha'$ *where* $\alpha' \neq \epsilon$ *and* $|u| = \mathrm{M}$ *and* $D \xrightarrow{u} D'(p'\alpha'\alpha, q'\beta'\delta)$.

1. *If* $D \models pX\alpha \sim q\beta\delta$ *then* $\mathsf{B}(D') \models p'\alpha' U\delta \sim q'\beta'\delta$.

2. *If* $n > \mathrm{M}$ *and* $D \models pX\alpha \sim_n q\beta\delta$ *and* $D' \models p'\alpha'\alpha \not\sim_{(n+1)-\mathrm{M}} q'\beta'\delta$ *then* $\mathsf{B}(D') \models p'\alpha' U\delta \not\sim_{(n+1)-\mathrm{M}} q'\beta'\delta$

Proof: If $D \models pX\alpha \sim q\beta\delta$ and $(U, \overline{D}_i)$ is an extended simple stack symbol for this true goal then $D_j \models p_j\alpha \sim U_j\delta$ for each D_j in $\overline{D}_i$. Assume $D \xrightarrow{u}$

[4]There may be more than one candidate as also $D \xrightarrow{w_i} D'_i(p_i\alpha, r'_i\gamma'_i\delta)$, but we only need to choose one.

$D'(p'\alpha'\alpha, q'\beta'\delta)$ and therefore $D' \models p'\alpha'\alpha \sim q'\beta'\delta$. We show by induction on $\mathrm{d}(E)$ that if $E \models p'_i\alpha'_i\alpha \sim q'\lambda'$ then $\mathsf{B}(E) \models p'_i\alpha'_iU\delta \sim q'\lambda'$. If $\mathrm{d}(E) = 0$ then the result follows because $p'_i\alpha'_i\alpha \sim p'_i\alpha'_iU\delta$. Otherwise there are two cases. $\mathsf{B}(E) \xrightarrow{a} F(p''\alpha''U\delta, q\lambda)$ and $\alpha'' \neq \epsilon$. By the construction $E \xrightarrow{a} E'(p''\alpha''\alpha, q\lambda)$ and $F = \mathsf{B}(E')$, and therefore by the induction hypothesis $\mathsf{B}(E') \models p''\alpha''U\delta \sim q\lambda$. The second case is similar except that $\alpha'' = \epsilon$. Therefore $\mathsf{B}(E) \xrightarrow{a} F(U_j\delta, q\lambda)$, and by the construction $E \xrightarrow{a} E'(p_j\alpha, q\lambda)$ and $F = D^c_j \circ E'$. However $D^c_j \models U_j\delta \sim p_j\alpha$ and $E' \models p_j\alpha \sim q\lambda$ and therefore by Fact 2.6 of the previous section $F \models U_j\delta \sim q\lambda$. From both of these cases it follows that $\mathsf{B}(E) \models p'_i\alpha'_iU\delta \sim q'\lambda'$.

Assume $n > M$ and $D \models pX\alpha \sim_n q\beta\delta$. Therefore it follows that $D_j \models p_j\alpha \sim_{(n+1)-\mathrm{M}} U_j\delta$ for each D_j in $\overline{D_i}$. We show by induction on $\mathrm{d}(E)$ that if $E \models p'_i\alpha'_i\alpha \not\sim_{(n+1)-t} q'\lambda'$ then $\mathsf{B}(E) \models p'_i\alpha'_iU\delta \not\sim_{(n+1)-t} q'\lambda'$ where $t \geq M$ and E is a subwitness of D'. If $\mathrm{d}(E) = 0$ then the result follows because $p'_i\alpha'_i\alpha \sim_{(n+1)-\mathrm{M}} p'_i\alpha'_iU\delta$. Otherwise there is a subwitness E' of E such that $E \xrightarrow{a} E'(p''\alpha''\alpha, q\lambda)$ and $E' \models p''\alpha''U\delta \not\sim_{n-t} q\lambda$. Therefore $\mathsf{B}(E) \xrightarrow{a} F(p''\alpha''U\delta, q\lambda)$. There are two cases. First $\alpha'' \neq \epsilon$ and $F = \mathsf{B}(E')$, and therefore by the induction hypothesis $\mathsf{B}(E') \models p''\alpha''U\delta \not\sim_{n-t} q\lambda$. Secondly $\alpha'' \neq \epsilon$ and $p'' = p_j$. By the construction $F = D^c_j \circ E'$. However $D^c_j \models U_j\delta \sim_{(n+1)-\mathrm{M}} p_j\alpha$ and $E' \models p_j\alpha \not\sim_{n-t} q\lambda$. Clearly $\mathrm{d}(E') < \mathrm{d}(D^c_j)$. Therefore using Fact 2.8 of the previous section $F \models U_j\delta \not\sim_{n-t} q\lambda$. In both these cases it therefore follows that $\mathsf{B}(E) \models p'_i\alpha'_iU\delta \not\sim_{(n+1)-t} q'\lambda'$. □

There is a symmetric definition of when $(U, \overline{D_i})$ is an extended simple stack symbol for $D \models q\beta\delta = pX\alpha$. Each witness D_j in $\overline{D_i}$ has root $(U_j\delta, p_j\alpha)$. The goal $D' \models q'\beta'\delta = p'\alpha'\alpha$ is then reduced to the balanced subgoal $\mathsf{B}(D') \models q'\beta'\delta = p'\alpha'U\delta$. Here the same notation is used for the transformation: it appeals to the symmetric iterative construction where, for example, there is the symmetric witness $E \circ D^c_j$ in case 2. Completeness and soundness of this subgoal reduction is justified by the symmetric version of Proposition 2.

6 Canonical recursive stack symbols

Bounding imbalance is not enough for showing decidability. The sizes of the "tails" δ may continually grow. The next step in the argument is a mechanism for eliminating them. It is at this point that we appeal to recursive stack symbols. The balanced goal $D \models p\alpha\delta = q\beta\delta$ can be reduced to a subgoal $D' \models p\alpha V = q\beta V$ where V is a recursive stack symbol. The mechanism involves constructing V from a subsidary family of goals $D'_i \models p'_i\alpha'_i\delta = q'_i\beta'_i\delta$ where $1 \leq i \leq k+1$, each with the same tail as the original goal (and where $k = |\mathsf{P}|$). As with Proposition 2 of the previous section care must be taken with its exact formulation in order to obtain a suitable witness D' from the witnesses D and $\overline{D'_i}$. As this is a central insight for the decision procedure, the desired result is built in stages.

Proposition 1 *If $p\alpha\delta \sim q\beta\delta$ where α, $\beta \neq \epsilon$ and do not contain recursive symbols then there is a recursive stack symbol $V \stackrel{\text{def}}{=} (r_1\lambda_1 V, \ldots, r_k\lambda_k V)$ such that*

1. *$p\alpha V \sim q\beta V$ and*
2. *$p_j\delta \sim r_j\lambda_j\delta$ for all $j : 1 \leq j \leq k$*

Proof: Assume $p\alpha\delta \sim q\beta\delta$ and assume α, $\beta \neq \epsilon$ and do not contain recursive stack symbols. A "canonical" recursive symbol V is constructed in stages as follows. First let $V^0 \stackrel{\text{def}}{=} (p_1 V^0, \ldots, p_k V^0)$. Clearly 2 holds for V^0. Next assume that V^i has been constructed and $V^i \stackrel{\text{def}}{=} (r_1\lambda_1 V^i, \ldots, r_k\lambda_k V^i)$ and 2 holds for V^i. If 1 also holds, $p\alpha V^i \sim q\beta V^i$, then V^i is the required recursive stack symbol V. Otherwise $p\alpha V^i \not\sim q\beta V^i$ and therefore there is a least $n > 0$ such that $p\alpha V^i \not\sim_n q\beta V^i$. Without loss of generality, assume that $p\alpha V^i \stackrel{a}{\longrightarrow} p'\alpha' V^i$ and $p'\alpha' V^i \not\sim_{n-1} q'\beta' V^i$ for all transitions $q\beta V^i \stackrel{a}{\longrightarrow} q'\beta' V^i$. However $p\alpha\delta \sim q\beta\delta$. Because $p\alpha\delta \stackrel{a}{\longrightarrow} p'\alpha'\delta$ there is a transition $q\beta\delta \stackrel{a}{\longrightarrow} q'\beta'\delta$ with $p'\alpha'\delta \sim q'\beta'\delta$. Thus we have the situation $p'\alpha' V^i \not\sim_{n-1} q'\beta' V^i$ whereas $p'\alpha'\delta \sim q'\beta'\delta$. We keep repeating this construction for increasing j yielding pairs $p''\alpha'' V^i \not\sim_{n-j} q''\beta'' V^i$ and $p''\alpha''\delta \sim q''\beta''\delta$. One possibility is that for some $j > 0$ the choice of transitions "enters" V^i. That is we have pairs $r_l\lambda_l V^i \not\sim_{n-j} q''\beta'' V^i$ and $p_l\delta \sim q''\beta''\delta$ (and similarly if V^i is entered in the right process). By 2 we know that $r_l\lambda_l\delta \sim p_l\delta$, and therefore we can continue the construction with the pairs $r_l\lambda_l V^i \not\sim_{n-j} q''\beta'' V^i$ and $r_l\lambda_l\delta \sim q''\beta''\delta$. In this way we maintain the same heads in the processes. Therefore eventually we must reach the situation where the pairs have the form $p_l V^i \not\sim_{n-j} q''\beta'' V^i$ and $p_l\delta \sim q''\beta''\delta$ and $(V^i)_l$, the lth entry of V^i, is $p_l V^i$ (and similarly if $p_l V^i$ is the right process). At this point V^i is refined to V^{i+1} in such a way that 2 is maintained. The argument proceeds by cases on $q''\beta''$.

Case 1 is that $\beta'' \neq \epsilon$. Therefore we change the entry $(V^i)_l$ to $q''\beta'' V^{i+1}$ and the same for any other entry $(V^i)_{l'}$ such that $(V^i)_{l'} = (V^i)_l$. For all other entries of V^i we merely replace occurrences of V^i with V^{i+1}. The result is a refined recursive symbol V^{i+1}. Note it is well defined because β'' cannot contain recursive symbols (because the starting sequence β does not contain them). Moreover 2 is maintained because $p_l\delta \sim q''\beta''\delta$ and if $p_{l'}\delta \sim p_l\delta$ then also $p_{l'}\delta \sim q''\beta''\delta$.

Case 2 is that $\beta'' = \epsilon$. Consider the states p_l and q''. Clearly $p_l \neq q''$ (for otherwise $p_l V^i \sim_{n-j} q'' V^i$). Therefore $p_l < q''$ or $q'' < p_l$, where $<$ is the ordering on states. Assume the first, the other case is similar. Assume q'' is $p_{l'}$. We update the entry $(V^i)_{l'}$ to $p_l V^{i+1}$ and the same for any other entry $(V^i)_{l''}$ such that $(V^i)_{l''} = (V^i)_{l'}$. For all other entries of V^i we replace occurrences of V^i with V^{i+1}. The result is a refined recursive symbol V^{i+1}. Note that we maintain the requirement on a recursive symbol V that if $V_{j'} = p_j V$ then $p_j \leq p_{j'}$. Moreover 2 is maintained because $p_l\delta \sim q''\delta$.

A sequence of recursive symbols $V^0, \ldots, V^i, \ldots$ is constructed, each member of which refines previous entries. However at each stage $i \geq 0$ exactly one entry of

the form $(V^i)_j = p_j V^i$ is directly updated (and others indirectly). Therefore after stage k it is not possible to directly update another entry. The final part of the argument is to show that 1 must hold for some stage V^i for $0 \leq i \leq k$. Suppose it fails for all $i < k$. It is now an easy argument to show that $p\alpha V^k \sim q\beta V^k$ using the fact that $p\alpha\delta \sim q\beta\delta$ and that V^k obeys 2. If $p\alpha V^k \not\sim_n q\beta V^k$ for some $n > 0$ then we build the sequence of pairs $p''\alpha''V^k \not\sim_{n-j} q''\beta''V^k$ and $p''\alpha''\delta \sim q''\beta''\delta$, as above, for increasing j. But now we cannot reach a situation where α'' or β'' is ϵ and the appropriate entry for V^k is $p''V^k$ or $q''V^k$. That means we must reach a situation where $p''\alpha''$ has an a-transition but $q''\beta''$ does not (or vice-versa) where α'' and β'' are not ϵ. But this contradicts that $p''\alpha''\delta \sim q''\beta''\delta$. □

The recursive stack symbol V is said to be canonical for $p\alpha\delta \sim q\beta\delta$ when given by the construction in the proof of this result. There is also a "depth" associated with V. At each stage i of the construction before V is reached there is the recursive symbol V^i and a least index n_i such that $p\alpha V^i \not\sim_{n_i} q\beta V^i$. The depth of V for $p\alpha\delta \sim q\beta\delta$ is the sum of the indices n_i appealed to in the construction.

The next result extends the notion of canonical recursive stack symbol to more than one true goal with the same tail.

Proposition 2 *Assume V is canonical for $p\alpha\delta \sim q\beta\delta$. If $p'\alpha'\delta \sim q'\beta'\delta$ and α', $\beta' \neq \epsilon$ and do not contain recursive symbols then there is a recursive stack symbol $U \stackrel{\text{def}}{=} (r'_1\lambda'_1 U, \ldots, r'_k\lambda'_k U)$ which refines V such that*

1. *$p\alpha U \sim q\beta U$ and*
2. *$p'\alpha' U \sim q'\beta' U$ and*
3. *$p_j\delta \sim r'_j\lambda'_j\delta$ for all $j : 1 \leq j \leq k$*

Proof: The construction extends the proof of Proposition 1. Assume $V \stackrel{\text{def}}{=} (r_1\lambda_1 V, \ldots, r_k\lambda_k V)$ is canonical for $p\alpha\delta \sim q\beta\delta$. Let $U^0 \stackrel{\text{def}}{=} (r_1\lambda_1 U^0, \ldots, r_k\lambda_k U^0)$. If $p'\alpha' U^0 \sim q'\beta' U^0$ then the result is proved with $U = V$. Otherwise there is a least n such that $p'\alpha' U^0 \not\sim_n q'\beta' U^0$ and so U^0 is refined to U^1 using the technique in the proof of Proposition 1. The refinement preserves $p\alpha U^1 \sim q\beta U^1$, see Fact 2 of section 3. Thus U^0 is continually refined until the result holds, as in the proof of Proposition 1. □

The construction can be extended to a family of goals with the same tail. Because a recursive stack symbol can only be refined at most k times, assume $l \leq k + 1$, and the family

$$\{p'_i\alpha'_i\delta \sim q'_i\beta'_i\delta \ : \ 1 \leq i \leq l\}$$

where each α'_i and $\beta'_i \neq \epsilon$ and does not contain recursive stack symbols. There is a canonical recursive stack symbol for this family. First one constructs a canonical

V for the initial member of the family $p'_1\alpha'_1\delta \sim q'_1\beta'_1\delta$ as in Proposition 1. Next one refines this to a possibly new V for $p'_2\alpha'_2\delta \sim q'_2\beta'_2\delta$ as in Proposition 2. This symbol is then refined for further members of the family, using Proposition 2, with result $V \stackrel{\text{def}}{=} (r_1\lambda_1 V, \ldots, r_k\lambda_k V)$ such that for each $i \leq l$, $p'_i\alpha'_i V \sim q'_i\beta'_i V$ and for each $j \leq k$, $p_j\delta \sim r_j\lambda\delta$. Again there is a depth associated with V, which is the sum of the indices n_i appealed to when refining V^i into V^{i+1}.

The final result before introducing witnesses is that there are only finitely many different canonical recursive stack symbols for families of goals which have the same heads but different tails.

Proposition 3 *Let $l \leq k+1$ and let $\{(p'_i\alpha'_i, q'_i\beta'_i) : 1 \leq i \leq l\}$ be an indexed family of heads where each $\alpha'_i, \beta'_i \neq \epsilon$ and does not contain recursive stack symbols. Let Δ be a family of sequences of stack symbols such that for each $\delta \in \Delta$, for all $i : 1 \leq i \leq l$, $p'_i\alpha'_i\delta \sim q'_i\beta'_i\delta$. There is a finite family $\mathcal{V}$ of recursive stack symbols such that for each $\delta \in \Delta$ there is a $V \in \mathcal{V}$ which is canonical for the family $p'_i\alpha'_i\delta \sim q'_i\beta'_i\delta$, $1 \leq i \leq l$.*

Proof: The finite family $\mathcal{V}$ is built iteratively starting from the initial pair $(p'_1\alpha'_1, q'_1\beta'_1)$. First let $V^0_\Delta \stackrel{\text{def}}{=} (p_1 V^0_\Delta, \ldots, p_k V^0_\Delta)$. At subsequent stages there is a finite family $V^i_{\Delta_1}, \ldots, V^i_{\Delta_{n_i}}$ where $\Delta_1, \ldots, \Delta_{n_i}$ is a finite partition of Δ (which may be an infinite set) such that if $\delta \in \Delta_j$ and $V^i_{\Delta_j} \stackrel{\text{def}}{=} (r_1\lambda_1 V^i_{\Delta_j}, \ldots, r_k\lambda_k V^i_{\Delta_j})$ then $p_n\delta \sim r_n\lambda_n\delta$ for $1 \leq n \leq k$. For the next stage consider the family $V^i_{\Delta_j}$. If for all j, $p'_1\alpha'_1 V^i_{\Delta_j} \sim q'\beta'_1 V^i_{\Delta_j}$ then consider the next pair $(p'_2\alpha'_2, q'_2\beta'_2)$, and continue refining the family of recursive stack symbols. Otherwise there is at least one stack symbol $V^i_{\Delta_j}$ such that there is a least n with $p'_1\alpha'_1 V^i_{\Delta_j} \not\sim_n q'_1\beta'_1 V^i_{\Delta_j}$. However for all $\delta \in \Delta_j$ we know that $p'_1\alpha'_1\delta \sim q'_1\beta'_1\delta$. Without loss of generality $p'_1\alpha'_1 V^i_{\Delta_j} \stackrel{a}{\longrightarrow} p''_1\alpha''_1 V^i_{\Delta_j}$ and for all transitions $q'_1\beta'_1 V^i_{\Delta_j} \stackrel{a}{\longrightarrow} q''_1\beta''_1 V^i_{\Delta_j}$, $p''_1\alpha''_1 V^i_{\Delta_j} \not\sim_{n-1} q''_1\beta''_1 V^i_{\Delta_j}$. For each $\delta \in \Delta_j$, $p'_1\alpha'_1\delta \stackrel{a}{\longrightarrow} p''_1\alpha''_1\delta$, and therefore for each such δ there is a transition $q'_1\beta'_1\delta \stackrel{a}{\longrightarrow} q''_1\beta''_1\delta$ such that $p''_1\alpha''_1\delta \sim q''_1\beta''_1\delta$. However the set $\{q''\beta'' : q'_1\beta'_1 \stackrel{a}{\longrightarrow} q''\beta''\}$ is finite: assume it is $\{q''_{11}\beta_{11}, \ldots, q''_{1m}\beta_{1m}\}$. Therefore there is a finite partition of $\Delta_j = \Delta_{j1}, \ldots, \Delta_{jm}$ such that $p''_1\alpha''_1 V^i_{\Delta_j} \not\sim_{n-1} q''_1\beta''_{1s} V^i_{\Delta_j}$ and for all $\delta \in \Delta_{js}$, $p''_1\alpha''_1\delta \sim q''_{1s}\beta''_{1s}\delta$. The argument is now repeated for each Δ_{js} until $V^i_{\Delta_j}$ is refined as in the proof of Proposition 1. However here the recursive symbol $V^i_{\Delta_j}$ will be refined to a family $V^{i+1}_{\Delta_{j1}}, \ldots, V^{i+1}_{\Delta_{jm}}$ where each $V^{i+1}_{\Delta_{js}}$ refines $V^i_{\Delta_j}$. There can only be k refinements, and at each stage each recursive stack symbol is refined into a finite family, and so the result follows. □

Proposition 3 is an important finiteness result. Given a family of heads $\{(p'_i\alpha'_i, q'_i\beta'_i) : 1 \leq i \leq l\}$ where $l \leq k+1$, there are finitely many canonical recursive stack symbols $\{V^1, \ldots, V^n\}$ such that for any tail δ if for all i, $p'_i\alpha'_i\delta \sim q'_i\beta'_i\delta$ then there is a V^j which is canonical for this family of true goals. Consequently we can extend the notion of "depth" to families of heads. The depth associated

with the family $\{(p'_i\alpha'_i, q'_i\beta'_i) : 1 \leq i \leq l\}$ is the maximum of the depths of the V^j as given by the construction in Proposition 3. There is no claim that this value is computable, only that it exists[5].

The next stage of the argument is to include bisimulation witnesses, and goals which need not be true. Assume $l \leq k+1$ and assume a family of goals with a common tail

$$\begin{aligned} D'_1 &\models p'_1\alpha'_1\delta = q'_1\beta'_1\delta \\ &\vdots \\ D'_l &\models p'_l\alpha'_l\delta = q'_l\beta'_l\delta \end{aligned}$$

where each $\alpha'_i, \beta'_i \neq \epsilon$ and does not contain recursive stack symbols. We wish to define a "canonical" recursive stack symbol V for this family of goals by "reading off" its entries from the witnesses $D'_1, \ldots, D'_l$. We will also need to know which witnesses justify the entries of V. Following the construction of the previous section for simple stack symbols, an extended recursive stack symbol is a pair $(V, \overline{D}_i)$ where $\overline{D}_i$ is a family of justifying witnesses: if $V_i = r_i\lambda_i V$ then witness D_i has root $(p_i\delta, r_i\lambda_i\delta)$. An extended recursive stack symbol is built iteratively. It also has an associated depth. The base case is $(V^0, \overline{D^0}_i)$ with depth 0 where $V^0 \stackrel{\text{def}}{=} (p_1V^0, \ldots, p_kV^0)$ and for each j, D^0_j is the identity witness $I(p_j\delta, p_j\delta)$.

Assume $s \leq l$ and $(V^j, \overline{D^j}_i)$ with depth d_j is canonical for the family of goals $D'_n \models p'_n\alpha'_n\delta = q'_n\beta'_n\delta$ when $n < s \leq l$. Consequently if $(V^j)_i = r_i\lambda_i V^j$ then D^j_i is a witness with root $(p_i\delta, r_i\lambda_i\delta)$. The next goal in the family is $D'_s \models p'_s\alpha'_s\delta = q'_s\beta'_s\delta$. An iterative subconstruction on D'_s with respect to $(V^j, \overline{D^j}_i)$ is now defined. Both D'_s and $(V^j, \overline{D^j}_i)$ may be updated. The updating of D'_s is a tree which is not a proper witness but contains well formed subwitnesses. Assume $D'_{s0} = D'_s$ and D'_{sm} with respect to $(V^{j'}, \overline{D^{j'}}_i)$ with depth $d_{j'}$ is the current pair in the iteration, $j' \geq j$. Consider any subwitness $E(p_i\delta, q\lambda\delta)$ or $E(q\lambda\delta, p_i\delta)$ of the tree D'_{sm} which has maximal depth and where E is not the identity witness I. Let d be $\mathrm{d}(D'_s) - \mathrm{d}(E)$. There are three cases to examine:

1. $p_i = q$ and $\lambda = \epsilon$. D'_{sm} is updated to $D'_{s(m+1)}$ by replacing E with the identity witness $I(p_i\delta, p_i\delta)$ of depth 0 and the construction is repeated for $D'_{s(m+1)}$ with respect to $(V^{j'}, \overline{D^{j'}}_i)$.

2. $(V^{j'})_i = r_i\lambda_i\delta_i$ and either $r_i \neq p_i$ or $\lambda_i \neq \epsilon$. D'_{sm} is updated to $D'_{s(m+1)}$ as follows. If E has root $(p_i\delta, q\lambda\delta)$ then E is replaced with $(D^{j'}_i)^c \circ E$. If E has root $(q\lambda\delta, p_i\delta)$ then it is replaced with $E \circ D^{j'}_i$. The construction is repeated for $D'_{s(m+1)}$ with respect to $(V^{j'}, \overline{D^{j'}}_i)$.

[5]This turns out to be the reason why there is not a complexity bound on the decision method of this paper.

3. Otherwise $p_i \neq q$ or $\lambda \neq \epsilon$ and $D_i^{j'}$ is the identity witness $I(p_i\delta, p_i\delta)$. In which case $V^{j'}$ is refined to $V^{j'+1}$. Assume E has root $(q\lambda\delta, p_i\delta)$, as the other case is similar. There are the two subcases. First is that $\lambda \neq \epsilon$. Then $V^{j'}$ is refined to $V^{j'+1}$ as in the proof of Proposition 1, with $(V^{j'+1})_i = q\lambda V^{j'+1}$ and any other l' such that $(V^{j'})_{l'} = p_i V^{j'}$ is also updated, $V_{l'}^{j'+1} = q\lambda V^{j'+1}$. The witnesses are also updated $D_i^{j'+1}$ is E^c, and $D_{l'}^{j'+1} = D_{l'}^{j'} \circ E^c$. The rest of the entries are unchanged. The second case is that $\lambda = \epsilon$. Consider which is the least state p_i or q with respect to the ordering on states. Assume it is q which is $p_{m'}$. Therefore $V^{j'}$ is refined to $V^{j'+1}$ as in the proof of Proposition 1 with $(V^{j'+1})_i = p_{m'} V^{j'+1}$ and any other l' such that $(V^{j'})_{l'} = p_i V^{j'}$ is similarly updated. The witnesses are also updated $D_i^{j'+1}$ is E^c, and $D_{l'}^{j'+1} = D_{l'}^{j'} \circ E^c$. The rest of the entries are unchanged. The result is a refined extended recursive stack symbol $(V^{j'+1}, \overline{D^{j'+1}}_i)$ whose depth is $d_{j'+1} = d_{j'} + d$. The construction is repeated for D'_{sm} with respect to $(V^{j'+1}, \overline{D^{j'+1}}_i)$ with depth $d_{j'+1}$.

The subconstruction must terminate after a finite number of steps with the pair D'_{sm} and $(V^{j'}, \overline{D^{j'}}_i)$ with depth $d_{j'}$. There can be at most k refinements of the extended stack symbol. D'_s has a finite depth and the construction iterates down the depth (and at each stage $\mathrm{d}(D'_{si}) \leq \mathrm{d}(D'_s)$). When it terminates if D'_{sm}[6] has a subwitness of the form $E(p_i\delta, q\lambda\delta)$ or $E(q\lambda\delta, p_i\delta)$ then $p_i = q$ and $\lambda = \epsilon$ and E is the identity witness I. $(V^{j'}, \overline{D^{j'}}_i)$ is then canonical for the family of goals $D'_n \models p'_n\alpha'_n\delta = q'_n\beta'_n\delta$ when $n \leq s \leq l$. The outermost construction is then repeated for the next goal in the family. However if $s = l$ then the extended stack symbol $(V^{j'}, \overline{D^{j'}}_i)$ with depth $d_{j'}$ is said to be canonical for the whole family of goals.

Below are properties of the witnesses in a canonical extended stack symbol, which underpin completeness and soundness of the construction.

Proposition 4 *Assume $(V, \overline{D}_i)$ with depth d is canonical for the family of goals $D'_i \models p'_i\alpha'_i\delta = q'_i\beta'_i\delta$, where $1 \leq i \leq l$.*

1. *If $D'_i \models p'_i\alpha_i\delta \sim q'_i\beta'_i\delta$ for all i, and D_j in $\overline{D}_i$ has root $(p_j\delta, r_j\lambda_j\delta)$ then $D_j \models p_j\delta \sim r_j\lambda_j\delta$.*

2. *If $D'_i \models p'_i\alpha'_i\delta \sim_n q'_i\beta'_i\delta$ for all i and $n > d$ and D_j in $\overline{D}_i$ has root $(p_j\delta, r_j\lambda_j\delta)$ then $D_j \models p_j\delta \sim_{n-d} r_j\lambda_j\delta$.*

Proof: Initially 1 holds for the witnesses in the initial extended stack symbol $(V^0, \overline{D^0}_i)$. We show that it is preserved by the iterative subconstruction. Assume it is true for $(V^{j'}, \overline{D^{j'}}_i)$, and assume $D'_s \models p'_s\alpha'_s\delta \sim q'_s\beta'_s\delta$. Therefore every

[6]Note that D'_{sm} is not technically a witness. However when the tail δ is replaced throughout D'_{sm} with $V^{j'}$ it is then a witness, as we note later in the proof.

subwitness of D'_s has a true root. This property is preserved by the iteration for D'_{sm} with respect to $(V^{j'}, \overline{D^{j'}}_i)$: for instance, in case 2 of the subconstruction $E \models p_i\delta \sim q\lambda\delta$ (when E has root $(p_i\delta, q\lambda\delta)$, the symmetric case is similar) and $(D_i^{j'})^c \models r_i\lambda_i\delta \sim p_i\delta$ and therefore by Fact 2.6 of section 4 $(D_i^{j'})^c \circ E \models r_i\lambda_i\delta \sim q\lambda\delta$. Consequently if $(V^{j'}, \overline{D^{j'}}_i)$ is refined as in case 3 of the subconstruction it follows that $E \models q\lambda\delta \sim p_i\delta$ and therefore if $V_i^{j'+1} = q\lambda V^{j'+1}$ then $D_i^{j'+1} \models p_i\delta \sim q\lambda\delta$, and similarly for other updates to $(V^{j'}, \overline{D^{j'}}_i)$.

For the proof of 2 we show the following iterative property. If $(V^j, \overline{D^j}_i)$ with depth d_j is the current extended stack symbol and D_i^j has root $(p_i\delta, r_i\lambda_i\delta)$ then $D_i^j \models p_i\delta \sim_{n-d_j} r_i\lambda_i\delta$. The result therefore follows. Initially, the property holds for $(V^0, \overline{D^0}_i)$ with depth 0 because $I \models p_i\delta \sim_n p_i\delta$. Assume it is true for $(V^{j'}, \overline{D^{j'}}_i)$ with depth $d_{j'}$ and assume $D'_s \models p'_s\alpha'_s\delta \sim_n q'_s\beta'_s\delta$ and $\mathrm{d}(D'_s) = d_s$. Consider the iteration for D'_{sm} with respect to $(V^{j'}, \overline{D^{j'}}_i)$. Every subwitness $E(p_i\delta, q\lambda\delta)$ or $E(q\lambda\delta, p_i\delta)$ whose depth is d' of D'_{sm} obeys $E \models p_i\delta \sim_{n-(d_{j'}+(d_s-d'))} q\lambda\delta$. This clearly holds for D'_{s0}. Assume a step of the subconstruction of case 2 when E has root $(p_i\delta, q\lambda\delta)$ and E has depth d' and $E \models p_i\delta \sim_{n-(d_{j'}+(d_s-d'))} q\lambda\delta$. By assumption $(D_i^{j'})^c \models r_i\lambda_i\delta \sim_{n-d_{j'}} p_i\delta$, and therefore $(D_i^{j'})^c \models r_i\lambda_i\delta \sim_{n-(d_{j'}+(d_s-d'))} p_i\delta$. From Fact 2.7 of section 4 it follows that $(D_i^{j'})^c \circ E \models r_i\lambda_i\delta \sim_{n-(d_{j'}+(d_s-d'))} q\lambda\delta$. When $(V^{j'}, \overline{D^{j'}}_i)$ is refined using the subwitness E with root $(p_i\delta, q\lambda\delta)$, the symmetric case is similar, it follows by the definition of depth that $D_i^{j'+1} \models p_i\delta \sim_{n-d_{j'+1}} q\lambda\delta$ because $d_{j'+1} = d_{j'} + (d_s - d')$, and similarly for any other update. □

The completeness property is subtle. Assume $(V, \overline{D}_i)$ with depth d is canonical for the family of true goals $\{D'_i \models p'_i\alpha'_i\delta = q'_i\beta'_i\delta \,:\, 1 \leq i \leq l\}$. Consider the new true goal $D \models p\alpha\delta = q\beta\delta$ with the same tail (where α and β are nonempty and do not contain recursive stack symbols). We would like to conclude with the subgoal $D' \models p\alpha V = q\beta V$, where D' is a witness built from D using $(V, \overline{D^i})$. The iterative subconstruction above is applied to D with respect to $(V, \overline{D^i})$. There are two possible outcomes. First is that $(V, \overline{D^i})$ is refined at some stage. In which case we extend the family of true goals with the extra true goal $D \models p\alpha\delta = q\beta\delta$, and refine $(V, \overline{D}_i)$. However this can only happen at most k times, when considering new true goals with the same tail. The other outcome is that $(V, \overline{D}_i)$ is not updated. The result of the iteration on D is the tree D'' which has the feature that if E is a subwitness of D'' with root $(p_i\delta, r\lambda\delta)$ or $(r\lambda\delta, p_i\delta)$ then E is the identity witness I. A simple operation on D'' can be performed: replace every tail δ throughout D'' with the recursive symbol V. Let $\mathsf{C}(D)$ with respect to $(V, \overline{D}_i)$ be the transformation of D involving these two stages, the construction of D'' and tail replacement. We assume that $\mathsf{C}(D)$ is only defined when the first stage does not update the extended stack symbol $(V, \overline{D}_i)$. The definition of

this transformation does not depend on the truth of any of the goals. Moreover, when defined $\mathsf{C}(D)$ is a well-formed witness. For instance, case 2 of the iterative subconstruction guarantees that if $pX \xrightarrow{a} p_j\epsilon$ and $E(pXV, r\lambda)$ is a subwitness of $\mathsf{C}(D)$ whose depth is at least 1 then there is a transition $E \xrightarrow{a} F(r_j\lambda_j V, r'\lambda')$ when $V_j = r_j\lambda_j V$.

If $\mathsf{C}(D) \models p\alpha V = q\beta V$ is a true goal then the goal $D \models p\alpha\delta = q\beta\delta$ reduces to it, and the tail δ, which may have arbitrary size, is thereby eliminated. Otherwise $\mathsf{C}(D) \models p\alpha V \not\sim q\beta V$. Because $\mathsf{C}(D)$ is a witness this means that it has a leaf $(p'\alpha' V, q'\beta' V)$ and $p'\alpha' V \not\sim q'\beta' V$. There is a similar leaf of D'', the tree constructed in the first stage of the transformation $\mathsf{C}(D)$, which has the form $(p'\alpha'\delta, q'\beta'\delta)$ and $p'\alpha'\delta \sim q'\beta'\delta$. Therefore V is not yet sufficiently canonical for the whole family of true goals and further refinement is necessary. There is a deeper witness $D' > D$ such that $D' \models p\alpha\delta \sim q\beta\delta$, and the iterative construction when applied to D' with respect to $(V, \overline{D}_i)$ refines this extended stack symbol.

There is also a finiteness property to highlight. Consider families G_δ of goals $\{D'_{\delta i} \models p'_i\alpha'_i\delta = q'_i\beta'_i\delta \,:\, 1 \le i \le l\}$ with the same heads, for each $\delta \in \Delta$. Assume that for each i and δ, $\mathrm{d}(D'_{\delta i}) \le d$. Clearly there are only finitely many different recursive stack symbols V in the family

$$\{(V, \overline{D}_i) \,:\, (V, \overline{D}_i) \text{ is canonical for some family } G_\delta,\ \delta \in \Delta\}$$

because any such $(V, \overline{D}_i)$ has depth at most kd: no subwitness involved in a refinement can have depth more than d and therefore each λ_j in an entry $V_j = r_j\lambda_j V$ has bounded size. The definition of when an extended recursive stack symbol $(V, \overline{D}_i)$ is canonical for a family of true goals G_δ is an "approximation" to when a recursive stack symbol V is canonical for the family without witnesses in the sense of Propositions 1, 2 and 3. If the depth d is sufficiently large and $(V, \overline{D}_i)$ is canonical to depth d for the family of true goals G_δ, $\{D'_{\delta i} \models p'_i\alpha'_i\delta = q'_i\beta'_i\delta\, 1 \le i \le l\}$, and G_δ contains appropriate witnesses then V is canonical for the family $\{p'_i\alpha'_i\delta \sim q'_i\beta'_i\delta \,:\, 1 \le i \le l\}$. To see this assume $p'_1\alpha'_1\delta \sim q'_1\beta'_1\delta$. Let $V^0 = (p_1V^0, \ldots, p_kV^0)$. If $p'_1\alpha'_1V^0 \sim q'_1\beta'_1V^0$ then there is a witness $D'_{\delta 1}$ for $p'_1\alpha'_1\delta \sim q'_1\beta'_1\delta$ such that $D'_{\delta 1}$ does not contain a subwitness $E(p_i\delta, q\lambda\delta)$ or $E(q\lambda\delta, p_i\delta)$ where $p_i \ne q$ or $\lambda \ne \epsilon$ and therefore $(V^0, \overline{D^0}_i)$ where each D^0_j is I is canonical for the true goal $D'_{\delta 1} \models p'_1\alpha'_1\delta = q'_1\beta'_1\delta$. Next assume $p'_1\alpha'_1V^0 \not\sim q'_1\beta'_1V^0$ and so there is a least n such that $p'_1\alpha'_1V^0 \not\sim_n q'_1\beta'_nV^0$. Therefore for any witness $D'_{\delta 1}$ for $p'_1\alpha'_1\delta \sim q'_1\beta'_1\delta$ of sufficient depth n there is a subwitness $E(p_i\delta, q\lambda\delta)$ or $E(q\lambda\delta, p_i\delta)$ where $p_i \ne q$ or $\lambda \ne \epsilon$ and $\mathrm{d}(D'_{\delta 1}) - \mathrm{d}(E) = n + 1$. V^0 is then refined to V^1 and $(V^0, \overline{D^0}_i)$ is refined to $(V^1, \overline{D^1}_i)$ for the goal $D'_{\delta 1} \models p'_1\alpha'_1\delta = q'_1\beta'_1\delta$, and so on.

Soundness is more straightforward than completeness. The exact formulation will become clearer in the next section.

Proposition 5 *Assume $(V, \overline{D}_i)$ with depth d is canonical for the family of goals $\{D'_i \models p'_i\alpha'_i\delta = q'_i\beta'_i\delta \,:\, 1 \le i \le l\}$, and for each i, $D'_i \models p'_i\alpha'_i\delta \sim_n q'_i\beta'_i\delta$.*

If $(d+m) \leq n+1$ *and* $D \models p\alpha\delta \not\sim_{(n+1)-(d+m)} q\beta\delta$ *and for each* $D_j \in \overline{D}_i$, $\mathrm{d}(D_j) \geq \mathrm{d}(D)$ *and* $\mathsf{C}(D)$ *is defined then* $\mathsf{C}(D)) \models p\alpha V \not\sim_{(n+1)-(d+m)} q\beta V$.

Proof: Assume $(V, \overline{D}_i)$ is canonical for the family $\{D'_i \models p'_i\alpha'_i\delta = q'_i\beta'_i\delta : 1 \leq i \leq l\}$, and for each i, $D'_i \models p'_i\alpha'_i\delta \sim_n q'_i\beta'_i\delta$. By Proposition 4.2 when $n > d$ it follows that for each $D_j \in \overline{D}_i$ with root $(p_j\delta, r_j\lambda_j\delta)$ that $D_j \models p_j\delta \sim_{n-d} r_j\lambda_j\delta$. Suppose $d+m \leq n+1$ and let $s = (n+1)-(d+m)$ and assume $D \models p\alpha\delta \not\sim_s q\beta\delta$ and $\mathsf{C}(D)$ is defined, but assume that $\mathsf{C}(D) \models p\alpha V \sim_s q\beta V$. We show this is impossible. Without loss of generality $D \xrightarrow{a} D'(p'\alpha'\delta, q'\beta'\delta)$ and $D' \models p'\alpha'\delta \not\sim_{s-1} q'\beta'\delta$. There are two cases. First $\alpha', \beta' \neq \epsilon$ and so by the iterative construction $\mathsf{C}(D) \xrightarrow{a} \mathsf{C}(D')$ and $\mathsf{C}(D') \models p'\alpha' V \sim_{s-1} q'\beta' V$. Without loss of generality assume for the second case that $\alpha' = \epsilon$ and $p' = p_j$. If $p' = q'$ and $\beta' = \epsilon$ then D' is I which is impossible. So either $p' \neq q'$ or $\beta' \neq \epsilon$. Assume $\beta' \neq \epsilon$ as the other case is similar. Therefore $V_j = r_j\lambda_j V$ and $\mathsf{C}(D) \xrightarrow{a} E$ and $E \models r_j\lambda_j V \sim_{s-1} q'\beta' V$. However we know that $D_j \models p_j\delta \sim_{s-1} r_j\lambda_j\delta$ and because $\mathrm{d}(D^j) \geq \mathrm{d}(D)$, $D^c_j \circ D' \models r_j\lambda_j\delta \not\sim_{s-1} q'\beta'\delta$ by Fact 2.7 of section 4. Moreover $E = \mathsf{C}(D^c_j \circ D')$ by the iterative construction. The argument is repeated for increasing j. There are subwitnesses D'' where $D'' \models p''\alpha''\delta \not\sim_{s-j} q''\beta''\delta$ but $\mathsf{C}(D) \models p''\alpha'' V \sim_{s-j} q''\beta'' V$. This leads to a contradiction when $j = s$. □

7 Tableaux

The proof of decidability is completed with a proof system for demonstrating bisimulation equivalence of processes. The proof system is goal directed and consists of two kinds of rules, "local" and "conditional". Local rules have the form

$$\frac{\text{Goal}}{\text{Subgoal}_1 \ \ldots \ \text{Subgoal}_n}\ C$$

where Goal is what currently is to be proved and the subgoals are what it reduces to, provided the side condition C holds. A conditional rule has the form

$$\frac{\begin{array}{c}\text{Goal}_1\\ \vdots\\ \text{Goal}_j\\ \vdots\\ \text{Goal}\end{array}}{\text{Subgoal}}\ C$$

where Goal is the current goal to be shown and there is a single subgoal to which it reduces provided that the goals Goal_1, ..., Goal_j occur above Goal on the path between it and the root (the starting goal) and provided that the side condition C holds.

$$\text{EXT}\ \frac{D \models p\alpha = q\beta}{D' \models p\alpha = q\beta}\ D' > D$$

$$\text{UNF}\ \frac{D \models p\alpha = q\beta}{D_1 \models p_1\alpha_1 = q_1\beta_1 \ldots D_n \models p_n\alpha_n = q_n\beta_n}\ C$$

where C is the condition

1. $\mathrm{d}(D) \geq 1$, and
2. $\{D' : D \xrightarrow{a} D' \text{ for } a \in \mathsf{A}\} = \{D_1, \ldots, D_n\}$

Figure 1: Simple tableau rules

There is also the important notion of when a current goal counts as final. Final goals are classified as either "successful" or "unsuccessful". A *tableau* proof for Goal is a finite proof tree whose root is Goal and all of whose leaves are successful final goals, and all of whose inner subgoals are the result of an application of one of the rules.

A goal has the form $D \models p\alpha = q\beta$ where D is a bisimulation witness for $(p\alpha, q\beta)$ to some depth d. D acts as an oracle which "determinizes" the development of the proof. We assume that we start with an initial goal $D \models p\alpha = q\beta$ where $\mathrm{d}(D) = 0$ and α and β do not contain auxiliary stack symbols.

The simple rules are presented in figure 1. The rule EXT allows one to extend the depth of a witness. The second rule UNF (unfold) reduces a goal $D \models p\alpha = q\beta$ to all its immediate subgoals as in the witness D. UNF obeys local completeness and soundness. Completeness is that if the goal is true then so are all the subgoals. This is clear from the definition of $D \models p\alpha \sim q\beta$ (and that $d(D) \geq 1$). Soundness is that if all the subgoals are true then so is the goal, or equivalently if the goal is false then so is at least one of the subgoals. A finer version uses approximants, which provide a measure of how false a goal is. Consider the smallest n such that $D \models p\alpha \not\sim_n q\beta$. In the case of UNF if the goal is false at $n+1$ then at least one of the subgoals is false at n. Soundness also clearly holds for EXT. However the statement of completeness is different, namely that there are correct applications of it, see Fact 2.1 of section 4.

The conditional rules are given in figure 2. The BAL rules introduce balanced subgoals and CUT cuts the common tail and replaces it in the subgoal with a recursive stack symbol. Completeness for BAL is that if the premise goals (those above the subgoal) are true then so is the subgoal, which follows from Proposition 2.1 and its symmetric version of section 5. The statement of completeness for CUT is that there are correct applications of it. The extended recursive stack

BAL(L)

$$\begin{array}{lcll} D & \models & pX\alpha = q\beta\delta & \\ \vdots & & \vdots & C_1 \\ D' & \models & p'\alpha'\alpha = q'\beta'\delta & \\ \hline D'' & \models & p'\alpha' U\delta = q'\beta'\delta & \end{array}$$

BAL(R)

$$\begin{array}{lcll} D & \models & q\beta\delta = pX\alpha & \\ \vdots & & \vdots & C_1 \\ D' & \models & q'\beta'\delta = p'\alpha'\alpha & \\ \hline D'' & \models & q'\beta'\delta = p'\alpha' U\delta & \end{array}$$

where C_1 is

1. $\mathrm{d}(D) \geq \mathrm{M}$ and $|\beta| = \mathrm{M}+1$ and β does not contain auxiliary stack symbols.
2. There are precisely M applications of UNF between the top goal and the bottom goal, and no application of any other rule.
3. D'' is $\mathsf{B}(D')$ using $(U, \overline{D}_i)$ which is an extended simple stack symbol for the initial goal.

CUT

$$\begin{array}{lcll} D'_1 & \models & p'_1\alpha'_1\delta = q'_1\beta'_1\delta & \\ \vdots & & \vdots & \\ D'_l & \models & p'_l\alpha'_l\delta = q'_l\beta'_l\delta & \\ \vdots & & \vdots & C_2 \\ D & \models & p\alpha\delta = q\beta\delta & \\ \hline D' & \models & p\alpha V = q\beta V & \end{array}$$

where C_2 is

1. $l \leq k+1$ and no α'_i, β'_i, α, β contains recursive stack symbols.
2. $(V, \overline{D}_i)$ with depth d is canonical for $\{D'_i \models p'_i\alpha'_i\delta = q'_i\beta'_i\delta \,:\, 1 \leq i \leq l\}$.
3. There are at least d applications of UNF between the goal $D'_l \models p'_l\alpha'_l\delta = q'_l\beta'_l\delta$ and $D \models p\alpha\delta = q\beta\delta$: there may also be applications of BAL but there are no applications of EXT.
4. $\mathsf{C}(D)$ with respect to $(V, \overline{D}_i)$ is defined, and $D' = \mathsf{C}(D)$.

Figure 2: Conditional tableau rules

Successful final goals

$$\begin{array}{ll} & D \models p\alpha = q\beta \\ & \vdots \qquad\qquad \text{UNF} \\ & \vdots \qquad \text{at least once} \\ I \models p\alpha = p\alpha & D \models p\alpha = q\beta \end{array}$$

Unsuccessful final goals

$$D \models p\alpha = q\beta \;\text{ and }\; D \models p\alpha \not\sim_1 q\beta$$

Figure 3: Final goals

symbol $(V, \overline{D}_i)$ is canonical for the first l premises and $\mathsf{C}(D)$ is well-formed, and therefore there will be occasions when the subgoal follows as discussed in the previous section.

To understand soundness of the conditional rules, we need to examine global soundness of the proof system. If there is a successful tableau whose root is false then there is a path through the tableau within which each subgoal is false. The idea is refined using approximants. If the root is false then there is an offending path (of false goals) through the tableau within which the approximant indices decrease whenever rule UNF has been applied, and hence this would mean that a successful final goal is false (which is impossible). Soundness of the conditional rules is that if the premises are on an offending path then the subgoal preserves the falsity index of the goal immediately above it. In the case of BAL(R) assume that the offending path passes through the premise goals. There is a least n such that for the initial premise $D \models q\beta\delta \sim_n pX\alpha$ and $D \models q\beta\delta \not\sim_{n+1} pX\alpha$. As there are exactly M applications of UNF between the initial and final premise it follows that $D' \models q'\beta'\delta \sim_{n-\mathrm{M}} p'\alpha'\alpha$ because D' is a subwitness of D and $\mathrm{d}(D') = \mathrm{d}(D) - \mathrm{M}$. However as this is the offending path $D' \models q'\beta'\delta \not\sim_{(n+1)-\mathrm{M}} p'\alpha'\alpha$, and therefore by the symmetric version of Proposition 2.2 of section 5 $D'' \models q'\beta'\delta \not\sim_{(n+1)-\mathrm{M}} p'\alpha'U\delta$. The same argument shows soundness of BAL(L). There is a similar argument for CUT. Assume that for each i, $D'_i \models p'_i\alpha'_i\delta \sim_n q'_i\beta'_i\delta$ and that $D'_l \models p'_l\alpha'_l\delta \not\sim_{n+1} q'_l\beta'_l\delta$. There are $s \geq d$ applications of UNF between it and the final premise, and so if this is an offending path $D \models p\alpha\delta \not\sim_{(n+1)-s} q\beta\delta$. Because $(V, \overline{D}_i)$ has depth d and is canonical for the first l premises and $\mathsf{C}(D)$ is defined $D' \models p\alpha V \not\sim_{(n+1)-s} q\beta V$ by Proposition 5 of the previous section.

Final goals are presented in figure 3. There is just one case for an unsuccessful final goal which is the situation when the bisimulation witness has depth 0 and cannot be extended because $p\alpha \not\sim_1 q\beta$. A final goal is successful if it is either an identity or a repeat. An offending path of false goals with decreasing falsity

indices cannot include either kind of successful goal. Clearly it is not possible for $I \models p\alpha \not\sim_n p\alpha$ for any n. For the other case, suppose the offending path passes through $D \models p\alpha = q\beta$ twice. At the first instance there is an n such that $D \models p\alpha \sim_n q\beta$ and $D \models p\alpha \not\sim_{n+1} q\beta$, but as there is at least one application of UNF between the two goals this would imply that $D \models p\alpha \not\sim_n q\beta$ which is a contradiction.

The first main result is that a successful tableau for $D \models p\alpha = q\beta$ indeed constitutes a proof that $D \models p\alpha \sim q\beta$.

Theorem 1 *If there is a successful tableau for $D \models p\alpha = q\beta$ then $D \models p\alpha \sim q\beta$.*

Proof: Suppose there is a successful tableau for $D \models p\alpha = q\beta$ but $D \models p\alpha \not\sim q\beta$. Then there is a least n such that $D \models p\alpha \not\sim_n q\beta$. We construct an offending path of false goals through the tableau within which the approximant indices decrease whenever UNF is applied. But this is impossible, as we must reach a successful final goal because the tableau is finite. □

In particular Theorem 1 applies to an initial goal, $D \models p\alpha = q\beta$ where $\mathrm{d}(D) = 0$ and α and β do not contain auxiliary stack symbols. More intricate is the proof of the converse that if $D \models p\alpha \sim q\beta$ then there is a successful tableau for $D \models p\alpha = q\beta$. The general idea is that given a true goal one applies the tableau proof rules, preserving truth, according to the strategy described below. It is therefore not possible to reach an unsuccessful final goal. Thus the main issue is how to guarantee that there is a finite tableau proof.

The strategy is parameterised with a depth md which is the maximal depth of a witness. This measure is appealed to whenever EXT is applied where the witness in the subgoal will have depth md. However the application of EXT is severely restricted, as we shall see, and this guarantees that every tableau built is finite. If there is not a successful tableau for a particular value of md with all possible choices of witnesses then the construction is restarted with an increased value of md. However if the initial goal is true then, as will be shown, at some stage there is a successful tableau. We shall also control the number of simple auxiliary stack symbols in a process $p\alpha$ to be at most two, and we shall maintain that for any recursive stack symbol V if $V_i \stackrel{\mathrm{def}}{=} r\lambda V$ then there is at most one simple stack element in λ.

We start with a simple observation

(1) For any $m \geq 0$ and $\mathrm{md} \geq 0$ and recursive stack symbol V there are only finitely many different goals of the form $D \models p\alpha = q\beta$ where $\mathrm{d}(D) \leq \mathrm{md}$ and $|\alpha| \leq m$ and $|\beta| \leq m$ and where either both α and β do not contain recursive stack symbols or both only contain V (as their final element).

The size of an application of BAL is the size of the process $q\beta\delta$ in the initial premise of the rule (see Figure 2), and the application is said to use $q\beta\delta$. Next

a value S for a process $p\alpha$ is introduced. If $p\alpha$ does not have a recursive stack symbol then S is $\mathrm{M}^2+2\mathrm{M}+2$. If it contains V then S is $\mathrm{M}^2+2\mathrm{M}+2+\max\{|\lambda_i| : V_i = r_i\lambda_i V\}$. A process whose size is less than or equal to S is said to be "small". A goal is small if both processes within it are small. Otherwise a goal is large.

We now describe the strategy. First we consider all the situations where EXT is applied to a goal G equal to $D \models p\alpha = q\beta$

1. G is the initial goal.
2. G is small and the result of an application of UNF
3. G is the result of an application of CUT

The strategy starts with the initial true goal and applies EXT to it so that the resulting witness has depth md. UNF is then repeatedly applied. If goals are small then EXT is also applied.

BAL is only permitted if the size of its application is more than S. The result of a BAL contains the process $p'\alpha' U\delta$. We say that $p'\alpha'$ is a "head" of an application and $U_i\delta$ is a tail for each $i : 1 \leq i \leq k$. Assume it is an application of BAL(L) as follows.

$$\begin{array}{rl} & q\beta\delta \\ & \vdots \qquad \mathrm{BAL(L)} \\ (*)\ D'' \models p'\alpha' U\delta = & q'\beta'\delta \end{array}$$

Just one new simple stack symbol is introduced into the subgoal $(*)$. If δ contains at most one simple stack symbol then there are at most two in $p'\alpha' U\delta$. Moreover the sizes of the configurations in the subgoal are bounded in the size of $q\beta\delta$. If the size of $q\beta\delta$ is m then $m - \mathrm{M} \leq |\beta'\delta| \leq m + \mathrm{M}$ and $(m+1) - \mathrm{M} \leq |\alpha' U\delta| \leq m+1$. The strategy is to repeatedly apply BAL(L) wherever possible, and UNF otherwise.

However BAL(R) is permitted once the "tail" of an application of BAL(L) is exposed. Assume an application of BAL(L) using $q'\beta'\delta'$, see Figure 4, resulting in the subgoal with witness D_1. Between this result and the goal $D_2 \models U_i'\delta' = q_j\lambda$ there are no further applications of BAL(L), and $U_i'\delta'$ is a tail of the BAL application. BAL(R) is now permitted provided it uses a process below, and including, $U_i'\delta'$. BAL(R) is not permitted using a process from a goal above $D_2 \models U_i'\delta' = q_j\lambda$. BAL(R) is not enforced, for one can still apply BAL(L). The strategy is always to apply a BAL rule whenever it is permitted. If BAL(R) is applied then the strategy is to repeatedly apply BAL(R), and to use UNF otherwise. BAL(L) is then only permitted when a tail of an application of BAL(R) is the right hand process of a goal.

This strategy ensures that at most one new simple stack symbol is introduced in a goal. Repeatedly applying BAL(L) either updates this stack symbol on the

$$
\begin{array}{lll}
 & q'\beta'\delta' & \\
 & \vdots & \text{BAL(L)} \\
D_1 \models p'\alpha'U'\delta' & = q''\beta''\delta' & \\
 & \vdots & \text{UNFs and EXTs} \\
D_2 \models U'_i\delta' & = q_j\lambda & \\
 & \vdots & \\
D_3 \models p\gamma & = r\lambda' &
\end{array}
$$

Figure 4: A potential switch from BAL(L) to BAL(R)

left hand side or reintroduces one. BAL(R) is only permitted once the left hand configuration no longer contains the new simple stack symbol. There is a bound on the number of UNFs between an application of BAL and when the other BAL rule becomes permitted when goals remain large. Consider the application of BAL(L) in Figure 4 with result $D_1 \models p'\alpha'U'\delta' = q''\beta''\delta'$. Between this goal and $D_2 \models U'_i\delta' = q_j\lambda$ there are no applications of BAL(L). The maximum number of applications of UNF between these goals is $\mathrm{M}(\mathrm{M}+1)$. The size of α' is at most $\mathrm{M}+1$. Suppose it has the form $X'_1 \ldots X'_l$, where $l \leq \mathrm{M}+1$. For BAL(L) not to apply after M applications of UNF there must be a process $p_i X'_2 \ldots X'_l U'\delta'$. Within M applications of UNF from the goal with this process there must be a process $p'_i X'_3 \ldots X'_l U'\delta'$, for otherwise BAL(L) would apply. And so on. Within $\mathrm{M}(\mathrm{M}+1)$ applications of UNF therefore there must be a goal with process $U'_i\delta'$.

Moreover if there is a sequence of large goals and both BAL rules are permitted but do not apply then the size of the goals must be decreasing. If the initial goal in the sequence is $D \models pX_1 \ldots X_n\alpha = qY_1 \ldots Y_n\beta$ then within $n\mathrm{M}$ applications of UNF there is a subgoal $D' \models p_i\alpha = r\lambda$ and a subgoal $D'' \models r'\lambda' = q_i\beta$. If the sequence is sufficiently long then eventually there must be a small goal.

Next we consider when a CUT is applied. First note that there can not be an infinite path of goals containing infinitely many small goals. By observation (1) such a path must contain infinitely many final goals (repeat goals). So consider a sequence of large goals whose applications of BAL are all at least as large as m. Assume that the initial process used in a BAL in this sequence has size m and has the form $q\beta\delta_1\delta$ where $|\beta\delta_1| = \mathrm{M}^2+2\mathrm{M}+2$. Then it follows that every process used in an application of BAL has the form $r\lambda\delta$ with the same tail δ (and where therefore $|\lambda| \geq \mathrm{M}^2+2\mathrm{M}+2$). For suppose not, consider the first application of BAL where this condition fails. There are two possibilities pictured in Figure 5. In both cases the first offending process used in an application of BAL is $r_2\lambda_2\delta'$. The previous application of BAL uses $r_1\lambda_1\delta$. The first case on the left is when there is not a switch of BAL, and for simplicity we assume that they are both applications of BAL(L). The second case on the right involves a switch which

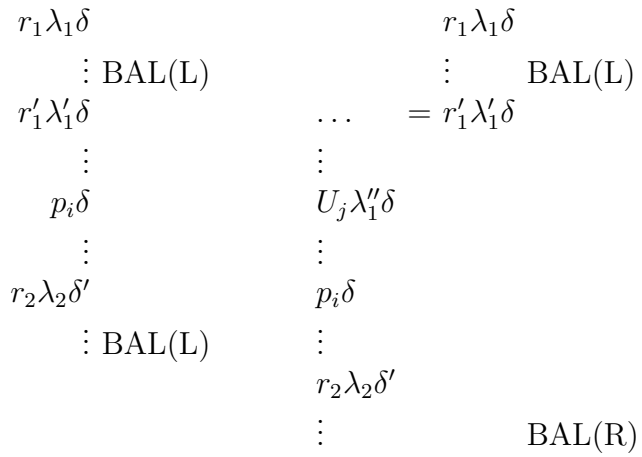

Figure 5: Two possibilities

for simplicity we have assumed is from BAL(L) to BAL(R). For the property to fail there must be a process which "enters" the tail δ, represented by $p_i\delta$, in both cases. However the processes $r_1\lambda_1\delta$ and $r_2\lambda_2\delta'$ have size at least m, and so $|\lambda_1| \geq \mathrm{M}^2 + 2\mathrm{M} + 2$. Consider the left case. Clearly $|\lambda_1'| \geq \mathrm{M}^2 + \mathrm{M} + 2$. Therefore there must be at least as many applications of UNF between it and $p_i\delta$. Because there are no applications of BAL it follows that BAL(R) is now permitted and therefore must apply somewhere between $p_i\delta$ and $r_2\lambda_2\delta'$ (as there must be at least $\mathrm{M}^2 + 2\mathrm{M} + 2$ applications of UNF which increase the size of the stack). Similarly for the right hand case $|U_j\lambda_1'| \geq \mathrm{M}^2 + \mathrm{M} + 3$. BAL(R) is permitted and therefore must apply somewhere between $p_i\delta$ and $r_2\lambda_2\delta'$ because again there must be at least $\mathrm{M}^2 + 2\mathrm{M} + 2$ applications of UNF which increase the size of the stack.

Consider a sequence of true large goals where all applications of BAL have size at least m. There are therefore no applications of EXT within this sequence (and therefore the length of the sequence is bounded because there can be at most md applications of UNF). Assume that the initial process used in a BAL is $q_1\beta_1\delta_1\delta$ where $|\beta_1\delta_1| = \mathrm{M}^2 + 2\mathrm{M} + 2$. The result of this BAL (say BAL(L)) is the true goal

$$D_1 \models p_1'\alpha_1'U^1\delta_1\delta = q_1'\beta_1'\delta_1\delta$$

This is to be the initial premise of a CUT with respect to the tail δ. The "heads" $p_1'\alpha_1'U^1\delta_1$, $q_1'\beta_1'\delta_1$ are bounded in size (no more than $\mathrm{M}^2 + 3\mathrm{M} + 3$). This bound is independent of the size of δ. Let $(V^1, \overline{D^1}_i)$ with depth d_1 be the canonical extended recursive stack symbol for this goal with respect to the tail δ. Note that no entry in V^1 contains more than a single simple stack element, namely U^1. Consider the first application of BAL after d_1 applications of UNF in this sequence. Suppose it uses $q_2\beta_2\delta_2\delta$ where δ is the same tail. The result of this

application, say a BAL(R) is the true goal

$$D_2 \models q_2'\beta_2'\delta_2\delta = p_2'\alpha_2'U^2\delta_2\delta$$

One possibility is that $(V^1, \overline{D^1}_i)$ is refined to $(V^2, \overline{D^2}_i)$ with depth d_2, which is then canonical for the pair of goals with witnesses D_1 and D_2. These two goals are potentially premises for CUT. Moreover if $(V^2)_i = r_i\lambda_i V^2$ then λ_i contains at most one simple stack symbol, either U^1 or U^2. The construction is then repeated. Consider the first application of a BAL after d_2 applications of UNF from the goal with witness D_2, and so on.

There can be at most k refinements of $(V^1, \overline{D^1}_i)$. Thus there is an $l \leq k+1$ and true goals $D_i' \models p_i'\alpha_i'\delta = q_i'\beta_i'\delta$ for $1 \leq i \leq l$ and an extended stack symbol $(V, \overline{D}_i)$ with depth d which is canonical for this family. And the result of the next application of BAL after d applications of UNF from the goal with witness D_l' is $D \models p\alpha\delta = q\beta\delta$ and $\mathsf{C}(D)$ with respect to $(V, \overline{D}_i)$ is defined. This means that $(V, \overline{D}_i)$ is canonical for the enlarged family of goals. The strategy is now to apply CUT with result $\mathsf{C}(D) \models p\alpha V = q\beta V$. EXT is applied to this result and the tableau construction continues with possibly further applications of CUT.

Any infinite path of goals with infinitely many applications of CUT must contain infinitely many repeats. There are only finitely many different heads $(p_1'\alpha_1', q'\beta_1')$ in an initial premise of a CUT. Moreover the heads of each subsequent premise of a CUT also has bounded size (the heads can not increase their size by more than md because that is the maximum number of UNFs which are allowed in a block). Therefore there are only finitely many different recursive stack symbols and finitely many different results of CUT.

The application of CUT does not necessarily preserve truth. The resulting goal $\mathsf{C}(D) \models p\alpha V = q\beta V$ may be false. In which case the tableau construction will fail. A second possibility for failure is that a sequence of goals in a block ends with a subgoal whose witness has depth 0 before CUT is applicable. The measure md is an upper bound on the number of applications of UNF within a block. When the construction fails for all choices of witnesses the depth md is increased, and the construction is restarted. However if md is sufficiently large then CUT will preserve truth because V in $(V, \overline{D}_i)$ will be canonical for the family $p_i'\alpha_i'\delta \sim q_i'\beta_i'\delta$ together with $p\alpha \sim q\beta$. Moreover, as we saw in the previous section, there are only finitely many canonical recursive stack symbols for families of goals with the same heads. This means that if the initial goal is true then there is a finite tableau proof for it. This concludes the decidability proof.

8 Conclusion

We have shown that bisimulation equivalence is decidable for pushdown processes. However no complexity measure is available because the decision procedure essentially relies on the introduction of canonical recursive stack symbols. Extra

Type	**Basic transitions**
Type -2	$R_1 \xrightarrow{a} R_2$
Type -1	$R_1 \xrightarrow{a} \beta$
Type 0	$\alpha \xrightarrow{a} \beta$
Type $1\frac{1}{2}$	$\alpha \xrightarrow{a} \beta$ where $\lvert\alpha\rvert = 2$ and $\lvert\beta\rvert > 0$
Type 2	$X \xrightarrow{a} \beta$
Type 3	$X \xrightarrow{a} \beta$ where $\lvert\beta\rvert \leq 1$

Figure 6: Caucal's hierarchy

insight is needed as to whether there is a bound on the size of a canonical recursive stack symbol in terms of the sizes of the heads α and β in the case of Proposition 1 of section 6.

Context-free grammars in Greibach normal form and pushdown processes are two means for generating the context-free languages (without the empty string). This class of languages is also generable using richer descriptions. Let N be a finite family of nonterminals and A a finite alphabet. We can then define a process as a sequence of nonterminals whose behaviour is determined by basic transitions together with a the following prefix rule.

$$\text{PRE if } \alpha \xrightarrow{a} \beta \text{ then } \alpha\delta \xrightarrow{a} \beta\delta$$

Figure 6 is Caucal's hierarchy of process descriptions according to how the family of basic transitions are specified. In each case we assume a finite family of rules. Type 3 captures finite-state systems. Type 2 captures BPA processes in Greibach normal form, and Type $1\frac{1}{2}$, in fact, captures pushdown processes. As noted in section 2 Type 0 also captures pushdown processes: Caucal proved that their transition graphs coincide (up to isomorphism) with pushdown processes [4]. For Type 0 and below in the hierarchy, in each case there are only finitely many basic transitions. One consequence of this is that the in and out degrees of vertices in their transition graphs are bounded. In the other two cases R_1 and R_2 are regular expressions over the nonterminals. The idea is that each rule $R_1 \xrightarrow{a} \beta$ stands for the possibly infinite family of basic transitions $\{\alpha \xrightarrow{a} \beta : \alpha \in L(R_1)\}$ and $R_1 \xrightarrow{a} R_2$ stands for the family $\{\alpha \xrightarrow{a} \beta : \alpha \in L(R_1) \text{ and } \beta \in L(R_2)\}$. For instance, a Type -1 rule of the form $X^*Y \xrightarrow{a} Y$ includes for each n the basic transition $X^nY \xrightarrow{a} Y$ and a Type -2 rule $X^*Y \xrightarrow{a} Z^*Y$ includes for each n and m the basic transition $X^nY \xrightarrow{a} Z^mY$. Consequently vertices of transition graphs of Type -1 processes may have infinite in degree but must have finite (and possibly unbounded) out degree. Vertices of Type -2 transition graphs may have infinite in and out degree.

Caucal's hierarchy is implicit in his work on understanding context-free graphs, and when the monadic second-order theory of a graph is decidable [4, 5, 6]. He

proves that the monadic second-order theory of a Type -2 graph is decidable [6]. With respect to language equivalence, there is no distinction between Type -2 and Type 2. With respect to bisimulation equivalence, there is a hierarchy as follows.

$$\text{Type}\,2 < \text{Type}\,1\frac{1}{2} = \text{Type}\,0 < \text{Type}\ -1 < \text{Type}\ -2$$

The result in this paper proves that bisimulation is decidable for Type 0 processes. This leaves as open questions whether it is also decidable for Type -1 and Type -2 processes.

A different perspective on the Caucal hierarchy uses pushdown automata with ϵ-transitions. Basic transitions have the form $pX \stackrel{a}{\longrightarrow} q\beta$ where $a \in \mathsf{A} \cup \{\epsilon\}$. However we assume the following disjointness property

$$\text{if } pX \stackrel{\epsilon}{\longrightarrow} q\beta \text{ then not}(\exists q\beta.\exists a \in \mathsf{A}.\, pX \stackrel{a}{\longrightarrow} q\beta)$$

A pushdown process is either "stable", unable to perform ϵ-transitions or "unstable", only able to perform ϵ-transitions. Our interest is in the collapsed graph of such a pushdown process, when ϵ-transitions are swallowed. It suffices to consider only stable processes and to define $p\alpha \stackrel{a}{\longrightarrow} q\beta$ if $p\alpha \stackrel{a}{\longrightarrow} p'\alpha' \stackrel{\epsilon^*}{\longrightarrow} q\beta$, as expected.

Various constraints can be placed on the basic transitions of these pushdown automata.

$\epsilon-$determinism : if $pX \stackrel{\epsilon}{\longrightarrow} q\beta$ and $pX \stackrel{\epsilon}{\longrightarrow} r\gamma$ then $q = r$ and $\beta = \gamma$
$\mathsf{A}-$determinism : if $a \in \mathsf{A}$ and $pX \stackrel{a}{\longrightarrow} q\beta$ and $pX \stackrel{a}{\longrightarrow} r\gamma$ then $q = r$ and $\beta = \gamma$
$\epsilon-$popping : if $pX \stackrel{\epsilon}{\longrightarrow} q\beta$ then $\beta = \epsilon$

If a PDA obeys the condition of ϵ-determinism then it has a normal form which also obeys ϵ-popping.

The classical DPDA problem, "do two DPDAs accept the same language?", is equivalent to the bismilarity problem on collapsed PDA graphs obeying ϵ-determinism and A-determinism. This was solved by Sénizergues [13, 14], and the proof was simplifed by the author [18],in effect, by viewing it as a bisimulation equivalence problem. Sénizergues generalised his proof to bisimulation equivalence between PDA obeying ϵ-determinism [15, 16]. The proof in this paper can also be extended to this case. A next step is to consider PDA which obey the weaker constraint ϵ-popping. It turns out that these coincide with Type -1 processes. Type -2 processes coincide with PDA without constraints. The proofs of these characterisations are very straightforward.

References

[1] Baeten, J., Bergstra, J., and Klop, J. (1993). Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of ACM*, **40**, 653-682.

[2] Burkart, O., Caucal, D., and Steffen, B. (1995). An elementary bisimulation decision procedure for arbitrary context-free processes. *Lecture Notes in Computer Science*, **969**, 423-433.

[3] Burkart, O., and Steffen, B. (1994). Pushdown processes: parallel composition and model checking. *Lecture Notes in Computer Science*, **836**, 98-113.

[4] Caucal, D. (1992). On the regular structure of prefix rewriting. *Theoretical Computer Science*, **106**, 61-86.

[5] Caucal, D. (1995). Bisimulation of context-free grammars and of pushdown automata. *CSLI Lecture Notes*, **53**, 85-106.

[6] Caucal, D. (1996). On infinite transition graphs having a decidable monadic theory. *Lecture Notes in Computer Science*, **1099**, 194-205.

[7] Christensen, S., Hüttel, H., and Stirling, C. (1995). Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, **121**, 143-148.

[8] Groote, J., and Hüttel, H. (1994). Undecidable equivalences for basic process algebra. *Information and Computation*, **115**, 354-371.

[9] Hopcroft, J., and Ullman, J. (1979). *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley.

[10] 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.

[11] Jancar, P. (1997). Decidability of bisimilarity for one-counter processes. *Lecture Notes in Computer Science*, **1256**, 549-559.

[12] Muller, D., and Schupp, P. (1985). The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, **37**, 51-75.

[13] Sénizergues, G. (1997). The equivalence problem for deterministic pushdown automata is decidable. *Lecture Notes in Computer Science*, **1256**, 671-681.

[14] Sénizergues, G. (1998). L(A) = L(B)? Tech. Report LaBRI, Université Bordeaux I, pp. 1-166. (Submitted to *Theoretical Computer Science*.)

[15] Sénizergues, G. (1998). Decidability of bisimulation equivalence for equational graphs of finite out-degree. *Procs. IEEE 39th FOCS*, 120-129.

[16] Sénizergues, G. (1998). $\Gamma(A) \sim \Gamma(B)$?. Manuscript, pp1-113.

[17] Stirling, C. (1998). Decidability of bisimulation equivalence for normed pushdown processes. *Theoretical Computer Science*, **195**, 113-131.

[18] Stirling, C. (1999). Decidability of DPDA equivalence. *Tech. Report LFCS-99-411*, University of Edinburgh, pp1-25. (Submitted to *Theoretical Computer Science*.)