

Language Theory and Infinite Graphs

Colin Stirling

School of Informatics, University of Edinburgh, UK

Abstract. Automata and language theory study finitely presented mechanisms for generating languages. A language is a family of words. A slight shift in focus is very revealing. Instead of grammars and automata as language generators, one views them as propagators of possibly infinite labelled transition graphs. This is our starting point for pushdown automata.

The main goal is to report on a proof of decidability of language equivalence for deterministic pushdown automata that uses both graph theoretic and combinatorial arguments.

Keywords. pushdown automata, language equivalence, decidability,

1. Pushdown Automata

The following four finite sets are ingredients of a pushdown automaton (PDA): states P , stack symbols S , alphabet A and basic transitions T . A basic transition has the form $pX \xrightarrow{a} q\alpha$ where $p, q \in P$, $X \in S$, $a \in A \cup \{\epsilon\}$ and $\alpha \in S^*$.

A configuration $p\beta$ of a PDA consists of a state and a sequence of stack symbols. The transitions of a configuration are defined by the following prefix rule where $\delta \in S^*$:
PRE if $pX \xrightarrow{a} q\alpha \in T$ then $pX\delta \xrightarrow{a} q\alpha\delta$. On input a the configuration $pX\delta$ in state p with X at the top of the stack changes to state q and α replaces X . Alternatively, with respect to a generational or process calculus perspective the configuration $pX\delta$ generates, or performs, a and becomes $q\alpha\delta$. If $a = \epsilon$ then configuration $pX\delta$ may change to $q\alpha\delta$ without reading an input or $pX\delta$ may become $q\alpha\delta$ silently without performing an observable action.

Throughout, we assume the following disjointness condition on T .

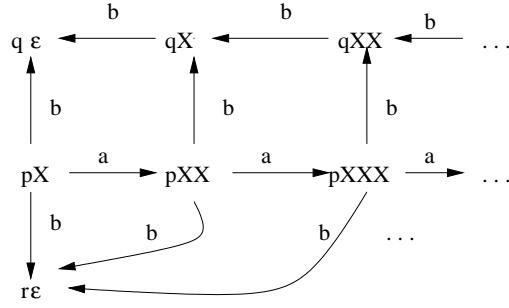
If $pX \xrightarrow{\epsilon} q\beta \in T$ and $pX \xrightarrow{a} r\lambda \in T$ then $a = \epsilon$

A configuration is *unstable*, only has ϵ -transitions, or *stable*, has no ϵ -transitions.

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

Example 1 Assume $P = \{p\}$, $S = \{X\}$, $A = \{a\}$ and $T = \{pX \xrightarrow{a} pXX\}$. $G(pX)$ is: $pX \xrightarrow{a} pXX \xrightarrow{a} pXXX \xrightarrow{a} \dots$. The transition $pXXX \xrightarrow{a} pXXXX$ follows by applying PRE to $pX \xrightarrow{a} pXX \in T$ with $\delta = XX$. \square

Example 2 Let $P = \{p\}$, $S = \{X, A, B\}$ and $A = \{a, b\}$. The basic transitions are $pX \xrightarrow{a} pAX$, $pA \xrightarrow{a} pAA$, $pB \xrightarrow{a} pAB$, $pX \xrightarrow{b} pBX$, $pA \xrightarrow{b} pBA$ and $pB \xrightarrow{b} pBB$. The graph $G(pX)$ is the full binary tree. \square

Figure 1. Collapsed graph $G^c(pX)$

Example 3 Let $P = \{p, q, r\}$, $S = \{X\}$, $A = \{a, b\}$. T is $pX \xrightarrow{a} pXX$, $pX \xrightarrow{b} r\epsilon$, $rX \xrightarrow{\epsilon} r\epsilon$, $pX \xrightarrow{b} q\epsilon$ and $qX \xrightarrow{b} q\epsilon$. The graph $G(pX)$ is

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

We use ϵ both as a transition label and for the empty stack sequence. \square

A PDA is presentable in normal form, up to isomorphism of transition graphs, where for each transition $pX \xrightarrow{a} q\alpha \in T$, the length of α , $|\alpha|$, is at most 2. A transition graph $G(p\alpha)$ has both bounded in and out degree. The out degree of a terminal configuration $p\alpha$ is 0 and the out degree of a configuration $pX\alpha$ is bounded by the number of basic transitions in T which have the form $pX \xrightarrow{a} q\beta$. The in degree of a configuration $p\alpha$ is bounded by the number of basic transitions in T which have the form $qY \xrightarrow{a} p\alpha'$, when α' is a prefix of α .

1.1. Collapsed graphs

A natural extension of transitions is to words $w \in A^*$. The extended transition $p\alpha \xrightarrow{w} q\beta$ represents that there is a w -path from configuration $p\alpha$ to configuration $q\beta$. For instance, $pXX \xrightarrow{ab} r\epsilon$ in the case of example 3. To capture word transitions, the *collapsed* graph of a PDA, which abstracts from ϵ -transitions, is defined. Consider just the *stable* configurations of example 3 above and transitions \xrightarrow{a} , $a \in A$, between them. This is the collapsed graph, without ϵ -transitions. For instance, its collapsed transition graph with stable root pX , written $G^c(pX)$, is depicted in figure 1. In a collapsed graph unstable configurations are removed, and $p\alpha \xrightarrow{a} q\beta$ if $p\alpha \xrightarrow{a} r\gamma(\xrightarrow{\epsilon})^*q\beta$. If a PDA does not contain ϵ -transitions then $G(p\alpha)$ is the same graph as $G^c(p\alpha)$.

A collapsed transition graph may have infinite in or out degree. The configuration $r\epsilon$ in figure 1 has infinite in degree because there are infinitely many transitions into it.

Example 4 Assume T is $rX \xrightarrow{a} pX$, $pX \xrightarrow{\epsilon} pXX$, $pX \xrightarrow{\epsilon} q\epsilon$ and $qX \xrightarrow{b} q\epsilon$. The graph $G(rX)$ is

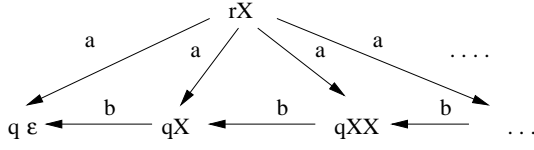


Figure 2. The collapsed graph $G^c(rX)$

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

The stable configurations in this graph are $\{rX, qX^n : n \geq 0\}$, and the collapsed graph $G^c(rX)$ is presented in figure 2. Infinite out degree, as illustrated by this example, depends on nondeterminism in the basic ϵ -transitions. \square

1.2. Subclasses of pushdown automata

The following are some important subclasses of pushdown automata.

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

There is a *normal form* in the case a PDA is ϵ -deterministic that ϵ -transitions only pop the stack: if $pX \xrightarrow{\epsilon} q\beta$ then $\beta = \epsilon$. By equivalent we mean that their collapsed graphs are isomorphic. Therefore, this is a normal form for a DPDA (*deterministic pushdown automaton*).

1.3. Decision questions

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

of monadic second-order logic, so model checking them on these infinite-state transition graphs is decidable. In practice, model checking these temporal logics appeal directly to automata (instead of monadic second-order logic).

Caucal also provided other characterisations of the graphs of pushdown automata. First he showed that the graphs of real-time PDA can be defined in terms of “pattern graphs” using deterministic graph grammars [7]. Secondly they are isomorphic to the transition graphs generated by 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 $\alpha \xrightarrow{a} \beta$ where α and β belong to N^* . The transition graph generated by a configuration $\delta \in N^*$ is determined by the basic transitions together with the prefix rule if $\alpha \xrightarrow{a} \beta$ then $\alpha\gamma \xrightarrow{a} \beta\gamma$ for any $\gamma \in N^*$. He also defined a more general class of graphs, the *prefix-recognisable* graphs [9], where the finite set of basic transitions are given by more general rules $E \xrightarrow{a} F$ where E and F are regular expressions over the alphabet of nonterminals. It turns out that these graphs coincide with the collapsed graphs of PDA. For instance, the graph of figure 2 is (isomorphic to) $G(X)$ when $X \xrightarrow{a} Y^*$ and $Y \xrightarrow{b} \epsilon$.

A third kind of decision question deals with *equivalence checking*. Classically, formal language theory views the language generable from a configuration as paramount. Therefore, the language equivalence problem is: given two configurations of a PDA¹ do they recognise the same language?

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

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

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

1.4. Bisimulation equivalence

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

Bisimulation equivalence is based on the existence of a relation between vertices of a transition graph which is preserved by transitions.

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

Definition 2 A binary relation R between vertices of transition graphs is a *bisimulation* relation provided that whenever $(E, F) \in R$, for all $a \in A$,

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

Definition 3 E is *bisimulation equivalent* to F , $E \sim F$, if there is a bisimulation relation containing (E, F) .

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

Definition 4 The family $\{\sim_n : n \geq 0\}$ between vertices of transition graphs is defined inductively as follows.

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

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

The converse of fact 1 is not in general true for collapsed graphs $G^c(p\alpha)$. However, it does hold for graphs which have finite out degree.

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

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

1.5. The DPDA equivalence problem

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

For pushdown automata, in general, the languages recognised under final state acceptance coincide with those recognised under empty stack acceptance. However, this is not true in the case of DPDA. The languages recognised by DPDA under final state acceptance are the *deterministic context-free* languages. The languages accepted under final state acceptance is a proper subset of these deterministic languages. However, we still work with empty stack acceptance because of the following. For any language L accepted by a DPDA configuration under final state acceptance, there is a DPDA configuration that accepts $L\$$ where $\$ \notin A$ is an end of word marker: the language $L\$ = \{w\$: w \in L\}$.

The DPDA equivalence problem was posed in the 1960s [12]: given two configurations $p\alpha$, $q\beta$ of a DPDA, is there an effective procedure for deciding whether $L(p\alpha) = L(q\beta)$? It was a celebrated open problem until it was solved positively by Geraud Sénizergues [24,26]. It seems that the notation of pushdown configurations, although

simple, is not rich enough to sustain a proof. Deeper algebraic structure needs to be exposed. Sénizergues's proof is primarily algebraic. This proof was simplified in [29,27]. However, these solutions to the problem involve two semi-decision procedures: one to show inequivalence ("find a smallest distinguishing word") the other to show equivalence (basically, using a *nondeterministic* proof procedure). Therefore, there is no complexity bound on the decision procedure.

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

2. Decidability of Equivalence of Simple Grammars

Surprisingly, the key to understanding the DPDA equivalence problem are *pushdown grammars* (PDGs) which are real-time single state PDA. A PDG has finite set ingredients stack symbols S , alphabet A and basic transitions T . A basic transition has the form $X \xrightarrow{a} \alpha$ where $X \in S$, $a \in A$ and $\alpha \in S^*$. A configuration β is a sequence of stack symbols. The prefix rule is: if $X \xrightarrow{a} \alpha \in T$ then $X\delta \xrightarrow{a} \alpha\delta$.

We assume that a PDG is in normal form. First, any transition $X \xrightarrow{a} \alpha \in T$ has the property that $|\alpha| \leq 2$. Second, a PDG is *normed*: for every $X \in S$ there is a word u such that $u \in L(X)$. We now show how to ensure this assuming a fixed total ordering $<$ on A .

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

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

It is easy to compute $w(X)$, if it exists, for each stack symbol X . Initially, stack symbols X with norm 1 are identified, and $w(X)$ is then defined, and then stack symbols with norm 2, and so on, until either $w(X)$ is calculated for each stack symbol X or X can not have a norm: the current largest norm is k and no stack symbol has norm between k and $2k + 1$.

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

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

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

In the worst case, M is exponential in $|S|$. The definition of norm extends to configurations of a PDG.

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

Clearly, $w(\epsilon) = \epsilon$ and $|w(\epsilon)| = 0$. Moreover, $w(X\alpha) = w(X)w(\alpha)$.

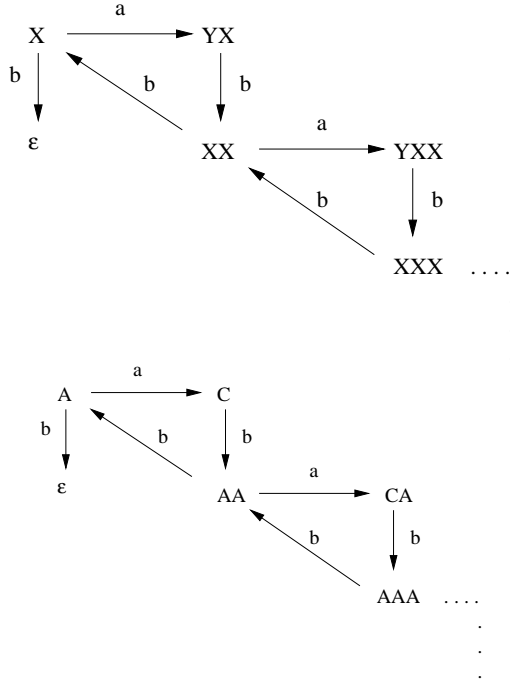


Figure 3. $G(X)$ and $G(A)$

As a first step towards solving the DPDA equivalence problem, we prove decidability of language equivalence for simple grammars, which was first shown by Korenjak and Hopcroft in 1966 using a method which is similar to that presented here [20].

2.1. Simple grammars

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

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

The recognition power of simple grammars is less than that of DPDA. For example, the language $L = \{a^n b^n : n > 0\} \cup \{a^n c : n > 0\}$ is generable by a DPDA configuration but not by a configuration of a simple grammar.

Example 1 Let T be $X \xrightarrow{a} YX$, $X \xrightarrow{b} \epsilon$, $Y \xrightarrow{b} X$, $A \xrightarrow{a} C$, $A \xrightarrow{b} \epsilon$ and $C \xrightarrow{b} AA$. The transition graphs $G(X)$ and $G(A)$ are pictured in figure 3. Here, $w(X) = b$ and $w(A) = b$. Because of the transition $Y \xrightarrow{b} X$, the symbol Y has norm 2 and $w(Y) = bb$. The symbol C has norm 3 because $C \xrightarrow{b} AA$, so $w(C) = bbb$. So the maximum norm $M = 3$. □

Configurations of a simple grammar are sequences of stack symbols. An extra configuration, the deadlocked configuration \emptyset , is added for the purpose of a useful notation²,

² \emptyset has an important role later when configurations are extended to sets of sequences of stack symbols $\{\alpha_1, \dots, \alpha_n\}$, and it is then genuinely the emptyset.

“ α after the word u ,” written $\alpha \cdot u$. The configuration $\alpha \cdot u$ is either the configuration β such that $\alpha \xrightarrow{u} \beta$ or it is the deadlocked configuration \emptyset . The configuration \emptyset is un-normed, its size, $|\emptyset|$, is 0 and $L(\emptyset) = \emptyset$. It is assumed that $\emptyset\alpha = \emptyset = \alpha\emptyset$. The inclusion of \emptyset guarantees that $(\alpha \cdot u)$ is always defined, and is unique because simple grammars are deterministic. In example 1, above, $(YX \cdot bab) = XXX$ and $(AA \cdot aa) = \emptyset$. Some obvious properties of “after” are now listed.

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

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

Fact 2 *If $u \in L(\alpha)$, then for any $v \in A^+$, $uv \notin L(\alpha)$.*

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

Fact 3 *$L(\alpha) = L(\beta)$ iff $\alpha \sim \beta$.*

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

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

R obeys the hereditary conditions for being a bismulation. □

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

2.2. Tableau proof rules

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

The initial tableau proof rule is UNF (unfold) in figure 4. The goal, $\alpha \doteq \beta$ reduces to the subgoal $(\alpha \cdot a) \doteq (\beta \cdot a)$ for each $a \in A$. The application of this simple rule is both “complete” and “sound”. Completeness is the property that if the goal, $\alpha \doteq \beta$, is true then so are all the subgoals, $(\alpha \cdot a_i) \doteq (\beta \cdot a_i)$.

Fact 4 *If $\alpha \sim \beta$, then for all $a \in A$, $(\alpha \cdot a) \sim (\beta \cdot a)$.*

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

Fact 5 *If $\alpha \sim_n \beta$ and $\alpha \not\sim_{n+1} \beta$, then for some $a \in A$, $(\alpha \cdot a) \not\sim_n (\beta \cdot a)$.*

Repeated applications of UNF continually reduce goals. For instance, the following tree of applications of UNF starts with goal $A \doteq X$ from example 1.

$$\begin{array}{c}
\text{UNF} \\
\frac{\alpha \dot{=} \beta}{(\alpha \cdot a_1) \dot{=} (\beta \cdot a_1) \dots (\alpha \cdot a_k) \dot{=} (\beta \cdot a_k)} \mathbf{A} = \{a_1, \dots, a_k\} \\
\text{BAL(L) and BAL(R)} \\
\begin{array}{ccc}
X\alpha \dot{=} \beta & & \beta \dot{=} X\alpha \\
\vdots & \mathbf{C} & \vdots \\
\alpha'\alpha \dot{=} \beta' & & \beta' \dot{=} \alpha'\alpha \\
\hline
\alpha'(\beta \cdot w(X)) \dot{=} \beta' & & \beta' \dot{=} \alpha'(\beta \cdot w(X))
\end{array}
\end{array}$$

where C is the condition

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

CUT

$$\frac{\alpha\delta \dot{=} \beta\delta}{\alpha \dot{=} \beta} \delta \neq \epsilon$$

Figure 4. Tableau proof rules

$$\begin{array}{c}
\frac{A \dot{=} X}{C \dot{=} YX} \text{UNF} \\
\frac{\emptyset \dot{=} \emptyset \quad \frac{AA \dot{=} XX}{CA \dot{=} YXX} \text{UNF}}{A \dot{=} X} \text{UNF}
\end{array}$$

In each application of UNF, the left subgoal is the result after a and the right subgoal is the result after b . If $\alpha' \dot{=} \beta'$ is a subgoal which is the result of m consecutive applications of UNF from the goal $\alpha \dot{=} \beta$, then there is a word u such that $|u| = m$ and $\alpha' = (\alpha \cdot u)$ and $\beta' = (\beta \cdot u)$. The word u is then said to be the associated word with this sequence of applications. In the example, above, $CA \dot{=} YXX$ is the result of 3 consecutive applications of UNF from $A \dot{=} X$. The associated word is aba .

A key idea is reduction of a goal to a “balanced” subgoal, and this is the role of the rules BAL(L), balance left, and BAL(R), balance right, in figure 4. Balance rules were introduced by Sénizergues [26] for the more general DPDA equivalence problem³: in his framework, they are the transformations T_B . The imbalance of a goal $\alpha \dot{=} \beta$ is the length of the largest prefix of α or β before they have a common tail. If $\alpha = \alpha_1\delta$ and $\beta = \beta_1\delta$ and the only common suffix of α_1 and β_1 is the empty sequence, then the imbalance

³We shall use the more general rules later.

between α and β is $\max\{|\alpha_1|, |\beta_1|\}$. If the imbalance between α and β is 0, then $\alpha = \beta$, so $\alpha \sim \beta$.

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

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

However, $|(Y_1 \dots Y_M \cdot u)| \leq 2M$, $|\alpha'| = |(X \cdot u)| \leq M+1$ and $|(Y_1 \dots Y_M \cdot w(X))| \leq 2M$ and therefore, the maximum imbalance of the subgoal is $3M + 1$.

Completeness of an application of BAL depends on the following two properties.

Fact 6 *If $X\alpha \sim \beta$ then $\alpha \sim (\beta \cdot w(X))$. If $\alpha \sim \beta$ then $\delta\alpha \sim \delta\beta$.*

Therefore, if $X\alpha \sim \beta$ and $\alpha'\alpha \sim \beta'$ then $\alpha'(\beta \cdot w(X)) \sim \beta'$. Soundness utilises the following properties.

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

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

In fact soundness is subtle. We need to bear in mind ‘‘global’’ soundness of the tableau construction. The main idea, as we shall see, is that if there is a successful tableau whose root is false then there is an offending path of false goals. This idea is refined using approximants. If a goal $\gamma \doteq \delta$ is false then there is a least n such that $\gamma \not\sim_n \delta$. The falsity index decreases through an application of UNF, as shown by fact 5. Consequently, if a successful tableau has a false root then there is an offending path of goals whose ‘‘falsity’’ index is decreasing whenever UNF is applied.

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

Fact 7 *If $X\alpha \sim_n \beta$ and $|w(X)| = m$ and $n > m$ and $\alpha'\alpha \not\sim_{(n+1)-m} \beta'$ and $|\alpha'| > 0$ then $\alpha'(\beta \cdot w(X)) \not\sim_{(n+1)-m} \beta'$.*

Successful final goals

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

Unsuccessful final goals

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

Figure 5. Final goals

The final rule in figure 4, CUT, allows common tails to be “cut” from a goal. Completeness and soundness of this rule follow from the next results.

Fact 8 If $\alpha\delta \sim \beta\delta$ then $\alpha \sim \beta$.

Proposition 2 If $\alpha\delta \not\sim_n \beta\delta$ then $\alpha \not\sim_n \beta$.

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

2.3. Successful tableaux

A missing ingredient in the tableau description is when a current goal is final. Final goals are either *successful* or *unsuccessful*, and are presented in figure 5. A successful final goal is either an identity or a repeat goal with at least one application of UNF between the repetition. An identity is clearly true: for the argument why a repeat is successful see theorem 2. An unsuccessful final goal $\alpha \doteq \beta$ is clearly false.

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

Definition 5 A *successful* tableau for $\alpha \doteq \beta$ is a finite proof tree with root $\alpha \doteq \beta$ and all of whose leaves are successful final goals. Otherwise a tableau is *unsuccessful*.

Example 3 We proved that $A \sim X$ in example 2. Below is a successful tableau for $A \doteq X$. The annotation explains which rules are applied. The initial goal $A \doteq X$ reduces to the subgoals $(A \cdot a) \doteq (X \cdot a)$ and $(A \cdot b) \doteq (X \cdot b)$. The second of these goals is a final successful goal, $\epsilon \doteq \epsilon$. There are two further applications of UNF to the first of these goals. The right subgoal $A \doteq X$ is a successful final goal because it is a repeat and there is at least one application of UNF between the repetition. BAL(L) is applied to the goal $CA \doteq YXX$ with initial premise $AA \doteq XX$. This fulfills the conditions of the application of BAL(L), $w(A) = b$ and so $|w(Y)| = 1$ and there is precisely 1 application of UNF between the two premises and no other rule is applied. The left configuration has

changed from AA to CA in the second goal, and therefore the rule sanctions replacement of the final occurrence of A with $(XX \cdot w(A))$ which is X . CUT is then applicable

$$\begin{array}{c}
 \frac{A \dot{=} X}{C \dot{=} YX} \text{UNF} \\
 \frac{C \dot{=} YX}{AA \dot{=} XX} \text{UNF} \\
 \frac{AA \dot{=} XX}{CA \dot{=} YXX} \text{UNF} \\
 \frac{CA \dot{=} YXX}{CX \dot{=} YXX} \text{BAL(L)} \\
 \frac{CX \dot{=} YXX}{C \dot{=} YX} \text{CUT}
 \end{array}$$

to the goal $CX \dot{=} YXX$ and the result is a final successful goal $C \dot{=} YX$ because it is a repeat. \square

In example 3 there is an application of BAL(L) to the goal $CA \dot{=} YXX$. However BAL(R) also applies to this goal. The rules allow tableaux that do not terminate, for instance, by only applying UNF. A simple grammar is deterministic, and therefore we would prefer that there is just one tableau for any goal. To achieve uniqueness of tableau, we appeal to the ordering on A that in an application of UNF, the subgoals are ordered relative to this ordering. In example 3 we assume that $a < b$.

Next we provide a strategy for applying tableau proof rules. The important issue is when to apply the BAL rules and with respect to which premises. We assume that a BAL rule applies to a goal using the premise above which is the closest. We then place the following priority on the tableau rules.

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

Given a goal one tries first to apply BAL(L), and if it is not applicable then one tries to apply BAL(R). Only if one of the BAL rules does not apply does one apply UNF. CUT is only applied after a BAL. The tableau of example 3 follows this strategy, and it is therefore the unique tableau for the initial goal $A \dot{=} X$.

2.4. Correctness of tableaux

Theorem 1 *There is a unique boundedly finite tableau for goal $\alpha \dot{=} \beta$.*

Proof: Given a goal and using the strategy for building a tableau it is clear that it must be unique. The main part of the proof is to show that any tableau for a starting goal $\alpha \dot{=} \beta$ has a bounded size. M is the maximum norm associated with the grammar. The size of a goal $\gamma \dot{=} \delta$ in a tableau is $\max\{|\gamma|, |\delta|\}$. Now we make a number of observations which prove the result. The first observation is straightforward. (1) For any $m \geq 0$ there are only boundedly many different goals whose size is at most m . (2) After an application of BAL followed by CUT the size of the resulting subgoal is no more than $3M + 1$. Hence any branch of a tableau contains only boundedly many applications of BAL, because otherwise goals will be repeated and a repeated goal is final.

A goal is small if its size is less than $(M^2 + M)$. Clearly there are only boundedly many small goals. Hence any branch of a tableau involving only small goals and only applications of UNF must be boundedly finite. If a goal is not small, we say it is “large”. The final property is to show that any branch in a tableau where there are only applications of UNF and no applications of BAL, but which contains large goals must be declining in size. More precisely, assume a large goal $\alpha \dot{=} \beta$ and without loss of generality assume that $|\alpha| \geq |\beta|$. The following property holds. (3) If there is a branch of the tableau starting from the large goal $\alpha \dot{=} \beta$ which contains $M^2 + M$ applications of UNF and no application of other rules, then there is a goal in this branch whose size is strictly smaller than that of $\alpha \dot{=} \beta$. \square

Theorem 2 [Soundness] *If the tableau for $\alpha \dot{=} \beta$ is successful then $\alpha \sim \beta$.*

Proof: Suppose there is a successful tableau for $\alpha \dot{=} \beta$ but $\alpha \not\sim \beta$. By theorem 1 this tableau is finite. There is a least approximant n such that $\alpha \not\sim_n \beta$. We construct an offending path of false goals through the tableau within which the approximant indices decrease whenever UNF is applied (by fact 5). The other rules (BAL and CUT) preserve the falsity indices. Because the tableau is finite and successful this means that the path of false goals must conclude with a final goal. But this is impossible. Clearly it is not possible to reach a final goal of the form $\gamma \dot{=} \gamma$ with $\gamma \not\sim_k \gamma$. Moreover it is not possible to reach a final goal which is a repeat. At the first instance there is a least k such that $\gamma \not\sim_k \delta$ and as there is at least one application of UNF between the goals this would imply that $\gamma \not\sim_{k-1} \delta$, which is a contradiction. \square

Theorem 3 [Completeness] *If $\alpha \sim \beta$ then the tableau for $\alpha \dot{=} \beta$ is successful.*

Proof: One just builds the tableau for $\alpha \dot{=} \beta$. The application of any rule preserves truth. Therefore it is not possible to reach an unsuccessful final goal, and by theorem 1 above the tableau for $\alpha \dot{=} \beta$ is finite, and therefore successful. \square

This is not an optimal decision procedure for simple grammars, see [6].

3. A Deeper Normal Form for DPDA

What are the key features that underpin decidability of language equivalence for simple grammars? First, language (or bisimulation) equivalence is a *congruence* with respect to stack prefixing: if $L(\alpha) = L(\beta)$ then $L(\delta\alpha) = L(\delta\beta)$. Second, language (or bisimulation) equivalence supports *cancellation* of postfix stacks: if $L(\alpha\delta) = L(\beta\delta)$ then $L(\alpha) = L(\beta)$. Congruence allows us to tear apart a configuration $\alpha'\alpha$ and replace its tail with a potentially equivalent configuration β with overall result $\alpha'\beta$, as used in the BAL rules. Cancellation, as used in CUT, has the consequence that goals have bounded size. (Soundness of congruence and cancellation depend on dual versions that employ approximants.)

We might try to extend the tableau proof rules to goals that involve DPDA configurations. The rule UNF works for *stable* configurations: the goal $p\alpha \dot{=} q\beta$ involving stable configurations reduces to subgoals $(p\alpha \cdot a_i) \dot{=} (q\beta \cdot a_i)$, $a_i \in A$, that involve stable configurations. The problem is the other rules. How can we tear apart DPDA (stable) configurations and replace parts with parts? Indeed, what is a *part* of a configuration?

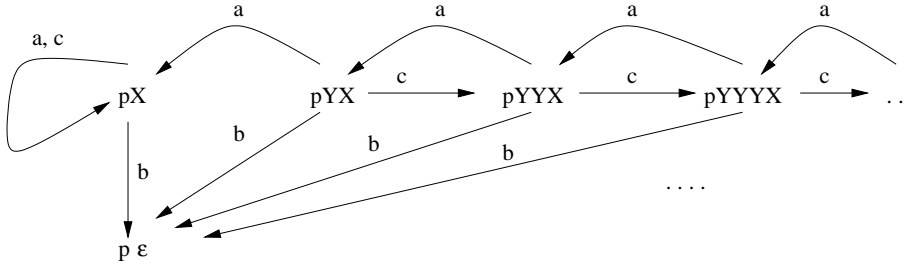


Figure 6. The graph $G^c(pYX)$

A configuration has a state as well as a stack. Clearly, $L(p\alpha) = L(q\beta)$ does not, in general, imply $L(p\delta\alpha) = L(q\delta\beta)$. Also, $L(p\alpha\delta) = L(q\beta\delta)$ does not, in general, imply $L(p\alpha) = L(q\beta)$.

Example 1 Consider the DPDA with transitions $pX \xrightarrow{a} pX$, $pX \xrightarrow{b} p\epsilon$, $pX \xrightarrow{c} pX$, $rX \xrightarrow{\epsilon} p\epsilon$, $pY \xrightarrow{a} p\epsilon$, $pY \xrightarrow{b} r\epsilon$, $pY \xrightarrow{c} pYY$ and $rY \xrightarrow{\epsilon} r\epsilon$. The collapsed graph $G^c(pYX)$ is pictured in figure 6. Although $L(pY^n X) = L(pY^m X)$ for every m and n , $L(pY^n) = L(pY^m)$ only if $n = m$. \square

This example also illustrates that there is not an obvious relation between the lengths of stacks of equivalent configurations.

Many of these problems disappear with pushdown grammars. Despite nondeterminism, stacking is a congruence with respect to pre and postfixing for language and bisimulation equivalence and both equivalences support pre and postfix cancellation.

Fact 1 $L(\alpha) = L(\beta)$ iff $L(\lambda\alpha\delta) = L(\lambda\beta\delta)$. $\alpha \sim \beta$ iff $\lambda\alpha\delta \sim \lambda\beta\delta$.

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

3.1. Strict deterministic grammars

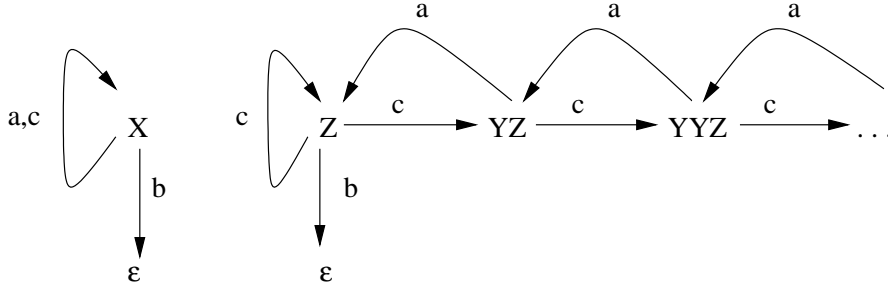
Strict deterministic grammars were introduced by Harrison and Havel [15], and further studied in [14,16]. They generate exactly the same languages as DPDA with empty stack acceptance.

A SDG, strict deterministic grammar, is a PDG with an extra component, an equivalence relation \equiv on S . The relation \equiv partitions the stack symbols S into disjoint subsets S_1, \dots, S_k : for each i , and pair of stack symbols $X, Y \in S_i$, $X \equiv Y$. We extend \equiv on S to a relation on S^* using the same notation.

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

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

Fact 2 $\alpha\beta \equiv \alpha$ iff $\beta = \epsilon$. $\alpha \equiv \beta$ iff $\delta\alpha \equiv \delta\beta$. If $\alpha \equiv \beta$ and $\gamma \equiv \delta$ then $\alpha\gamma \equiv \beta\delta$. If $\alpha \equiv \beta$ and $\alpha \neq \beta$ then $\alpha\gamma \equiv \beta\delta$. If $\alpha\gamma \equiv \beta\delta$ and $|\alpha| = |\beta|$ then $\alpha \equiv \beta$.

Figure 7. The graph $G(X)$ and $G(Z)$

Definition 2 The relation \equiv on S is *strict* if the following two conditions hold for $X \equiv Y$. (1) If $X \xrightarrow{a} \alpha \in T$ and $Y \xrightarrow{a} \beta \in T$, then $\alpha \equiv \beta$. (2) If $X \xrightarrow{a} \alpha \in T$ and $Y \xrightarrow{a} \alpha \in T$ then $X = Y$. An enhanced PDG is a *strict deterministic grammar*, SDG, if its relation \equiv is strict.

Example 2 $S = \{X, Y, Z\}$, $A = \{a, b, c\}$ and T comprises $X \xrightarrow{a} X$, $X \xrightarrow{b} \epsilon$, $X \xrightarrow{c} X$, $Y \xrightarrow{a} \epsilon$, $Y \xrightarrow{c} YY$, $Z \xrightarrow{b} \epsilon$, $Z \xrightarrow{c} Z$ and $Z \xrightarrow{c} YZ$. The graphs of X and Z are pictured in figure 7. $G(Z)$ is nondeterministic because there are two c -transitions from Z . Consider the following partition of S : $\{\{X\}, \{Y, Z\}\}$. This enhanced PDG is an SDG. An examination of its transitions reveals that only condition 1 of strictness needs to be checked for each pair of the three transitions $Y \xrightarrow{c} YY$, $Z \xrightarrow{c} Z$ and $Z \xrightarrow{c} YZ$. However, $YY \equiv Z$, $YY \equiv YZ$ and $Z \equiv YZ$. \square

We now examine some features of an SDG. If each set in the partition is a singleton, so only $X \equiv X$ for each X , then the SDG is *deterministic*, and is therefore a simple grammar. In this extreme case $\alpha \equiv \beta$ iff $\alpha = \beta$. Example 2 illustrate that when the partition involves larger sets then the SDG may exhibit nondeterminism. However, nondeterminism is “constrained”. For example, if $X \xrightarrow{a} \epsilon \in T$ and $X \equiv Y$ and $X \neq Y$, then there is not an a -transition $Y \xrightarrow{a} \beta \in T$. The strictness conditions generalise to all words $w \in A^*$ and to stack sequences.

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

Proof: The proof in both cases is by a routine induction on $|w|$. \square

3.2. Sum configurations

A configuration of a SDG is a sequence of stack symbols, α . We extend the notion of a configuration to sets of sequences of stack symbols, $\{\alpha_1, \dots, \alpha_n\}$. However, we shall write a set as a sum, $\alpha_1 + \dots + \alpha_n$ without repeated elements. Two sum configurations $\alpha_1 + \dots + \alpha_n$, and $\beta_1 + \dots + \beta_m$ are identical if they are the same set. A degenerate case is the empty sum, which we write as \emptyset . The language of a sum configuration is just the union of the languages of the components: $L(\alpha_1 + \dots + \alpha_n) = \bigcup \{L(\alpha_i) : 1 \leq i \leq n\}$. Therefore, $L(\emptyset) = \emptyset$. $L(X + Z)$, where $X + Z$ is from example 2, is $\{a^n b : n \geq 0\} \cup \{c^n c^m a^m b : n, m \geq 0\}$.

Only a subset of sum configurations are interesting.

Definition 3 $\beta_1 + \dots + \beta_n$ is *admissible*, if $\beta_i \equiv \beta_j$ for each i and j .

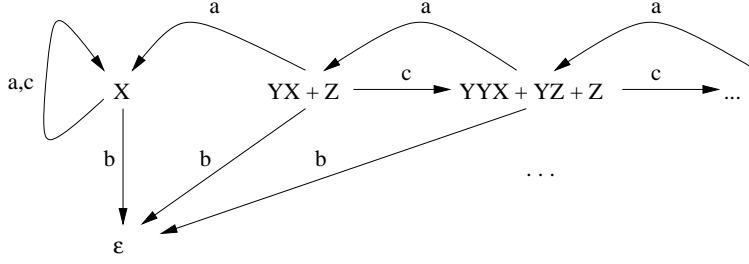


Figure 8. The graph $G^d(YX + Z)$

The empty sum, \emptyset , is therefore admissible. In [16] admissible configurations are called “associates”. Admissible configurations from example 2 include: XX , $ZZZ + ZZZY$, $YX + Z$, $Z + YZ$, $Z + YZ + YYZ$. The last example is admissible because $Z \equiv YZ$, $Z \equiv YYZ$ and $YZ \equiv YYZ$. The sum $X + Z$ is *not* admissible.

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

Fact 3 If $\alpha_1 + \dots + \alpha_n$ is admissible then $\{\beta_i^1 : \alpha_1 \xrightarrow{w} \beta_i^1\} \cup \dots \cup \{\beta_i^n : \alpha_n \xrightarrow{w} \beta_i^n\}$ is admissible for any word $w \in A^*$.

An application of fact 3 is *determinization* of a SDG where T is replaced by T^d . For each $X \in S$ and $a \in A$, the family of transitions $X \xrightarrow{a} \alpha_1, \dots, X \xrightarrow{a} \alpha_n \in T$ is determinized by replacing them with the single transition $X \xrightarrow{a} \alpha_1 + \dots + \alpha_n \in T^d$. The sum configuration $\alpha_1 + \dots + \alpha_n$ is admissible. For each X and a there is a unique transition $X \xrightarrow{a} \sum \alpha_i \in T^d$: if there is not an a transition for X in T then $X \xrightarrow{a} \emptyset \in T^d$. The prefix rule PRE for generating transitions is generalised to cover admissible configurations: if $X_i \xrightarrow{a} \alpha_1^i + \dots + \alpha_{n_i}^i$ for each $i : 1 \leq i \leq m$ then $X_1\beta_1 + \dots + X_m\beta_m \xrightarrow{a} (\alpha_1^1\beta_1 + \dots + \alpha_{n_1}^1\beta_1) + \dots + (\alpha_1^m\beta_m + \dots + \alpha_{n_m}^m\beta_m)$. If the antecedent sum configuration of PRE is admissible then so is the consequent configuration. The determinized transition graph $G^d(\alpha_1 + \dots + \alpha_n)$ is generated from the admissible configuration $\alpha_1 + \dots + \alpha_n$ using the determinized transitions T^d and the extended prefix rule.

Example 3 We reconsider example 2 where $S = \{X, Y, Z\}$, $A = \{a, b, c\}$ and the partition of $S = \{\{X\}, \{Y, Z\}\}$. T^d comprises $X \xrightarrow{a} X$, $Y \xrightarrow{a} \epsilon$, $Z \xrightarrow{a} \emptyset$, $X \xrightarrow{b} \epsilon$, $Y \xrightarrow{b} \emptyset$, $Z \xrightarrow{b} \epsilon$, $X \xrightarrow{c} X$, $Y \xrightarrow{c} YY$ and $Z \xrightarrow{c} YZ + Z$. The graph $G^d(YX + Z)$ is pictured in figure 8. It is not an accident that this graph is isomorphic to the graph of figure 6, as we shall see later. $YX + Z \xrightarrow{c} YYX + YZ + Z$ because $YX \xrightarrow{c} YYX$ and $Z \xrightarrow{c} YZ + Z$. All sum configurations in the graph are admissible. \square

3.3. Characterising DPDA

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

1. For every $p, q \in P$ and $X \in S$, introduce a stack symbol $[pXq] \in S_1$.

2. For transitions, the initial step is to define the following transitions for $a \in A$.
 - (a) If $pX \xrightarrow{a} q\epsilon \in \mathbb{T}$, then $[pXq] \xrightarrow{a} \epsilon \in \mathbb{T}_1$.
 - (b) If $pX \xrightarrow{a} qY \in \mathbb{T}$, then $[pXr] \xrightarrow{a} [qYr] \in \mathbb{T}_1$ for each $r \in P$.
 - (c) If $pX \xrightarrow{a} qYZ \in \mathbb{T}$, then $[pXr] \xrightarrow{a} [qYp'][p'Zr] \in \mathbb{T}_1$ for each $r, p' \in P$.
3. A stack symbol $[pXq] \in S_1$ is an ϵ -symbol, if $pX \xrightarrow{\epsilon} q\epsilon \in \mathbb{T}$. All ϵ -symbols are erased from the right hand side of any transition in \mathbb{T}_1 .
4. Next, the SDG is normalised by removing all unnormed stack symbols and useless transitions.
5. Finally, for elements of S_1 , if $p = r$ then $[pSq] \equiv [rSt]$ for any q, t .

Fact 4 If $w \in A^*$ and pX is stable in the DPDA, then $pX \xrightarrow{w} q\epsilon$ if, and only if, $[pXq] \xrightarrow{w} \epsilon$ in the SDG. The relation \equiv on S_1 is an equivalence relation and is strict.

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

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

Fact 5 $L(p\alpha) = L(\text{sum}(p\alpha))$

Moreover, the transition graph $G^c(p\alpha)$ is almost isomorphic to⁴ $G^d(\text{sum}(p\alpha))$. There can be $|P|$ co-roots, vertices of the form $q\epsilon$, in $G^c(p\alpha)$ in contrast with the single co-root, ϵ , in $G^d(\text{sum}(p\alpha))$.

Example 4 An example is the conversion of the DPDA of example 1. The stack symbols S_1 is $\{[pXp], [pXr], [pYp], [pYr], [rXp], [rXr], [rYp], [rYr]\}$. The transitions in \mathbb{T} are then converted.

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

There are two ϵ -stack symbols, $[rXp]$ and $[rYr]$. These are erased from the right hand side of any transition: the transition $[pYr] \xrightarrow{c} [pYr][rYr]$ is changed to $[pYr] \xrightarrow{c} [pYr]$. The stack symbols $[pXr]$, $[rYp]$, $[rXp]$, $[rXr]$ and $[rYr]$ are redundant and are removed. This reduces S_1 to the set $\{[pXp], [pYp], [pYr]\}$, and the transitions \mathbb{T}_1 to the following set

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

⁴In fact, is bisimulation equivalent to.

The partition is into the sets $\{\{[pXp]\}, \{[pYp], [pYr]\}\}$. Finally, the transitions T_1 are determined: the two c -transitions from $[pYr]$ are replaced by the single transition $[pYr] \xrightarrow{c} [pYr] + [pYp][pYr]$. The SDG is example 3 when $X = [pXp]$, $Y = [pYp]$ and $Z = [pYr]$. The configuration $pY Y X$ of the DPDA becomes the admissible configuration $[pYp][pYp][pXp] + [pYp][pYr] + [pYr]$ of the SDG, that generates the same language. \square

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

3.4. Heads, tails and extensions

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

Definition 4 The *size* of an admissible configuration $E = \beta_1 + \dots + \beta_n$, written $|E|$, is the length of its longest sequence, $\max\{|\beta_j| : 1 \leq j \leq n\}$. And $|\emptyset| = 0$.

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

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

Fact 6 If $E + F$ is admissible and $u \in L(E)$, then for any v , $uv \notin L(F)$. If $E + F$ is admissible, then $L(E) \cap L(F) = \emptyset$. $L(EF) = \{uv : u \in L(E) \text{ and } v \in L(F)\}$.

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

Definition 5 Assume $k \geq 1$ and $E = \{\beta'_1\delta_1, \dots, \beta'_m\delta_m\}$ and $|\beta'_i| = k$, or $|\beta'_i| < k$ and $\delta_i = \varepsilon$, for each $i : 1 \leq i \leq m$. If β_1, \dots, β_n are the distinct elements in $\{\beta'_1, \dots, \beta'_m\}$ and $H_l = \{\delta_j : \beta'_j = \beta_l\}$ for each $l : 1 \leq l \leq n$, then $E = \beta_1H_1 + \dots + \beta_nH_n$ is in *k-head form*.

Fact 7 If $E = \beta_1G_1 + \dots + \beta_nG_n$ is in *k-head form*, then $\beta_1 + \dots + \beta_n$ is admissible and each G_i is admissible and different from \emptyset .

Definition 6 $E = E_1G_1 + \dots + E_nG_n$ is in *head/tail form*, if the head $E_1 + \dots + E_n$ is admissible and at least one $E_i \neq \emptyset$, and each tail $G_i \neq \emptyset$.

Example 5 $E = YYYX + YYZ + YZ + Z$ is an admissible configuration of example 3. The partition of the stack symbols is $\{\{X\}, \{Y, Z\}\}$. E has 1-head form, $YG_1 + ZG_2$, where $G_1 = Y Y X + Y Z + Z$ and $G_2 = \varepsilon$. Also, E has 2-head form, $Y Y H_1 +$

$YZH_2 + ZH_3$, where $H_1 = YX + Z$ and $H_2 = H_3 = \varepsilon$. E cannot be presented as $YYG'_1 + YYG'_2 + YG'_3 + ZG'_4$: this is not a valid head/tail form because the head $YY + YY + Y + Z$ is not admissible ($YY \neq Y$) and it is not disjoint ($YY + YY$ is not a proper sum). \square

In the following, if a configuration E is written $E_1G_1 + \dots + E_nG_n$, then we assume that it fulfills the conditions of definition 6 of a head/tail form.

A key definition, used in the decision procedure later, is when one configuration *tail extends* another. Consider first configurations of a simple grammar $\alpha_1 = \beta_1\gamma$, $\alpha_2 = \beta_2\lambda\gamma$ and $\alpha_3 = \beta_3\delta\lambda\gamma$. Each α_i has possibly differing heads β_i and differing tails. However, α_2 's tail *extends* α_1 's (with λ) and α_3 's extends α_2 's with δ . Therefore, α_3 's tail extends α_1 's tail with the composition of λ and δ . We now extend the idea to admissible configurations.

Definition 7 If $E = E_1G_1 + \dots + E_nG_n$, $F = F_1H_1 + \dots + F_mH_m$ then F in its head/tail form is a *tail extension* of E in its head/tail form provided that for each $i : 1 \leq i \leq m$, $H_i = K_1^iG_1 + \dots + K_n^iG_n$. If F is a tail extension of E , then the associated extension e is the m -tuple $(K_1^1 + \dots + K_n^1, \dots, K_1^m + \dots + K_n^m)$ without the G_i s, and the size of e , written $|e|$, is $\max\{|K_j^i| : 1 \leq i \leq n \text{ and } 1 \leq j \leq m\}$ and the width of e is m , and F is said to extend E with e .

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

Proposition 2 If $E = E_1G_1 + \dots + E_lG_l$ and $E' = E'_1G'_1 + \dots + E'_mG'_m$ and $E'' = E''_1G''_1 + \dots + E''_nG''_n$ and E' extends E by $e = (J_1^1 + \dots + J_l^1, \dots, J_1^m + \dots + J_l^m)$ and E'' extends E' by $f = (K_1^1 + \dots + K_m^1, \dots, K_1^n + \dots + K_m^n)$, then E'' extends E by $ef = (H_1^1 + \dots + H_l^1, \dots, H_1^n + \dots + H_l^n)$ where $H_j^i = K_1^iJ_j^1 + \dots + K_m^iJ_j^m$ and $|ef| \leq |e| + |f|$.

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

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

Example 6 The following uses example 3. Let $E = YG_1 + ZG_2$ where $G_1 = X$ and $G_2 = \varepsilon$. $E' = YG'_1 + ZG'_2$ where $G'_1 = YX + Z$ and $G'_2 = \varepsilon$. $E'' = YG''_1 + ZG''_2$ where $G''_1 = YYX + YZ + Z$ and $G''_2 = \varepsilon$. E' extends E by $e = (Y + Z, \emptyset + \varepsilon)$ and E'' extends E' by $f = e = (Y + Z, \emptyset + \varepsilon)$. So, E'' extends E by $ef = (YY + (YZ + Z), \emptyset + \varepsilon)$. \square

3.5. Dynamic properties of head and tail forms

We adopt notation from section 2. For each $X \in S$, $w(X)$ is the unique shortest word $v \in A^+$ such that $X \xrightarrow{v} \varepsilon$. $w(E)$ is the unique shortest word for configuration E . M is the maximum norm of the SDG. E after u , written $E \cdot u$, is the unique F such that $E \xrightarrow{u} F$.

Fact 8 $(\alpha_1 + \dots + \alpha_n) \cdot u = (\alpha_1 \cdot u) + \dots + (\alpha_n \cdot u)$

Proposition 3 Assume $E = \beta_1 G_1 + \dots + \beta_n G_n$ is in k -head form, and each $\beta_j = X_1^j \dots X_{k_j}^j$. (1) If $(\beta_i \cdot u) = \varepsilon$, then for each $j \neq i$, $(\beta_j \cdot u) = \emptyset$ and $(E \cdot u) = G_i$. (2) If $(\beta_i \cdot u) = X_m^i \dots X_{k_i}^i$, then $(E \cdot u) = E_1 G_1 + \dots + E_n G_n$ where $E_i = (\beta_i \cdot u)$ and for $j \neq i$, either $E_j = \emptyset$ or $E_j = X_m^j \dots X_{k_j}^j$ and $X_1^j \dots X_{m-1}^j$ is the same sequence as $X_1^i \dots X_{m-1}^i$.

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

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

Proposition 4 Assume $E = E_1 G_1 + \dots + E_n G_n$. (1) If $(E_i \cdot u) = \varepsilon$, then for all $j \neq i$, $(E_j \cdot u) = \emptyset$ and $(E \cdot u) = G_i$. (2) If $(E_i \cdot u) \neq \emptyset$, then $(E \cdot u) = (E_1 \cdot u) G_1 + \dots + (E_n \cdot u) G_n$. (3) If $H_i \neq \emptyset$ for all $i : 1 \leq i \leq n$, then $E_1 H_1 + \dots + E_n H_n$ is a head/tail form. (4) If each $H_i \neq \emptyset$ and each $E_i \neq \varepsilon$ and for each j such that $E_j \neq \emptyset$, $H_j \sim_m G_j$, then $E \sim_{m+1} E_1 H_1 + \dots + E_n H_n$. (5) If each $H_i \neq \emptyset$ and for each j such that $E_j \neq \emptyset$, $H_j \sim G_j$, then $E \sim E_1 H_1 + \dots + E_n H_n$.

Proof: Part 1 is a simple generalisation of proposition 3 (1) and is proved in the same way. Part 2 follows from it. $E_1 G_1 + \dots + E_n G_n$ is a head/tail form, and, therefore, by definition each $G_i \neq \emptyset$ and $E_1 + \dots + E_n$ is admissible. Therefore, if each $H_i \neq \emptyset$, then $E_1 H_1 + \dots + E_n H_n$ is a head/tail form. For 4 assume $F = E_1 H_1 + \dots + E_n H_n$ and $E \not\sim_{m+1} F$. E and F have the same heads $E_1 + \dots + E_n$ and each $E_i \neq \varepsilon$. Therefore, there must be a word u with $0 < |u| \leq m + 1$ and $(E \cdot u) = G_i$ and $(F \cdot u) = H_i$ and $G_i \not\sim_{(m+1)-|u|} H_i$ which is a contradiction. The final part uses a similar argument. \square

Two configurations may have the same heads and different tails, or may have the same tails and different heads. If E has the head/tail form $E_1 G_1 + \dots + E_n G_n$ and F has a similar head/tail form $F_1 G_1 + \dots + F_n G_n$ involving the same tails⁵, then the imbalance between E and F , relative to this presentation, is $\max \{ |E_i|, |F_i| : 1 \leq i \leq n \}$. If the imbalance is 0, then they are the same. The next result establishes a key property of a combination of such configurations.

Proposition 5 Assume the following configurations. $E = E_1 G_1 + \dots + E_n G_n$, $F = F_1 G_1 + \dots + F_n G_n$, $E' = E_1 H_1 + \dots + E_n H_n$ and $F' = F_1 H_1 + \dots + F_n H_n$. If $E \sim_m F$ and $E' \not\sim_m F'$, then there is a word u , $|u| \leq m$, and an i such that either (1) or (2). (1) $(E' \cdot u) = H_i$ and $(F' \cdot u) = (F_1 \cdot u) H_1 + \dots + (F_n \cdot u) H_n$ and $(F' \cdot u) \neq \emptyset$ and $(E' \cdot u) \not\sim_{m-|u|} (F' \cdot u)$. (2) $(F' \cdot u) = H_i$ and $(E' \cdot u) = (E_1 \cdot u) H_1 + \dots + (E_n \cdot u) H_n$ and $(E' \cdot u) \neq \emptyset$ and $(E' \cdot u) \not\sim_{m-|u|} (F' \cdot u)$.

Proof: Assume $E \sim_m F$ and $E' \not\sim_m F'$. Therefore, there is a word u , $|u| = m$, and, without loss of generality, $(E' \cdot u) = \emptyset$ and $(F' \cdot u) \neq \emptyset$, by definition of \sim_m . However, $(E \cdot u) = \emptyset$ if, and only if, $(F \cdot u) = \emptyset$. Therefore, there must be a smallest prefix v

⁵Any pair of configurations E and F have a head/tail form involving the same tails: $E = EG$ and $F = FG$ when $G = \varepsilon$.

$$\begin{array}{c}
\text{UNF} \\
\hline
\frac{E \doteq F}{(E \cdot a_1) \doteq (F \cdot a_1) \quad \dots \quad (E \cdot a_k) \doteq (F \cdot a_k)} \quad \mathbf{A} = \{a_1, \dots, a_k\} \\
\text{BAL(R)} \\
\begin{array}{c}
F \doteq X_1 H_1 + \dots + X_k H_k \\
\vdots \\
F' \doteq E_1 H_1 + \dots + E_k H_k \\
\hline
F' \doteq E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k))
\end{array} \quad \mathbf{C} \\
\text{BAL(L)} \\
\begin{array}{c}
X_1 H_1 + \dots + X_k H_k \doteq F \\
\vdots \\
E_1 H_1 + \dots + E_k H_k \doteq F' \\
\hline
E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k)) \doteq F'
\end{array} \quad \mathbf{C}
\end{array}$$

where C is the condition

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

Figure 9. The tableau proof rules

of u such that either $(E' \cdot v) = H_i$ and $(F' \cdot v) = (F_1 \cdot v)H_1 + \dots + (F_n \cdot v)H_n$, or $(F' \cdot v) = H_i$ and $(E' \cdot v) = (E_1 \cdot v)H_1 + \dots + (E_n \cdot v)H_n$, and $(E' \cdot v) \not\sim_{m-|v|} (F' \cdot v)$. The result follows because $(E \cdot v) \sim_{m-|v|} (F \cdot v)$ and E has the same head as E' and F has the same head as F' and because $G_i, H_i \neq \emptyset, (E' \cdot v), (F' \cdot v) \neq \emptyset$. \square

4. Decidability of DPDA Equivalence

The procedure for deciding $E \sim F$ is a tableau proof system as in section 2. The proof rules, in figure 9, are generalisations of the unfold and balance rules for simple grammars. UNF reduces a goal $E \doteq F$ to subgoals $(E \cdot a) \doteq (F \cdot a)$ for each $a \in \mathbf{A}$. Because it is intended that there is a unique tableau associated with any goal, the subgoals are ordered by the ordering on \mathbf{A} . It is complete and sound.

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

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

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

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

As in section 2, if $E' \dot{=} F'$ is a subgoal given by m consecutive applications of UNF (and no other rule) to $E \dot{=} F$, then there is an *associated* word u such that $|u| = m$ and $E' = (E \cdot u)$ and $F' = (F \cdot u)$.

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

Definition 1 An application of BAL use F if F is the configuration in the initial goal of the rule, as in figure 9.

Fact 3 captures precisely the bounds of imbalance: the statement is for BAL(L), and its symmetric version covers BAL(R).

Fact 3 *Assume $E' \dot{=} F'$ is the result of BAL(L) using F . $|E'| \leq |F| + 2M + 1$ and $|F'| \leq |F| + M$. If $F = \beta_1 G_1 + \dots + \beta_n G_n$ is in m -head form and $m \geq M$, then $E' = E_1 G_1 + \dots + E_n G_n$ and each $|E_i| \leq m + 2M + 1$ and $F' = F_1 G_1 + \dots + F_n G_n$ and $|F_i| \leq m + M$.*

The BAL rules are sound and complete (compare section 2).

Fact 4 *If $X_1 H_1 + \dots + X_k H_k \sim F$ and $E_1 H_1 + \dots + E_k H_k \sim F'$, then $E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k)) \sim F'$. If $X_1 H_1 + \dots + X_k H_k \sim_{n+m} F$ and $E_1 H_1 + \dots + E_k H_k \not\sim_{n+1} F'$ and each $E_i \neq \epsilon$ and $m \geq \max\{|w(X_i)| : E_i \neq \emptyset\}$, then $E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k)) \not\sim_{n+1} F'$.*

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

It is intended that there be a unique tableau associated with any initial goal. So restrictions will be placed on which rule is to be applied when. First, as in section 2, the initial premise of a BAL is the one that is closest to the goal and, therefore, the one that involves the least number of applications of UNF. To resolve which rule should be applied, the following priority order is assumed: if BAL(L) is permitted, then apply BAL(L), if BAL(R) is permitted, then apply BAL(R) and otherwise, apply UNF. However, whether an application of BAL is permitted involves more than fulfillment of the side condition. It also depends on the previous application of a BAL.

Initially, either BAL is permitted provided that its side condition is true. If an application of BAL uses F , then the resulting goal contains the configuration $E_1(F \cdot w(X_1)) + \dots + E_k(F \cdot w(X_k))$. E_i is a ‘‘top’’ of the application of BAL and $(F \cdot w(X_i))$ is a ‘‘bottom’’. Assume an application of BAL(L). A subsequent application of BAL(L) is permitted provided the side condition of the rule is fulfilled. However, BAL(R) is not permitted until a bottom of the previous application of BAL(L) is exposed and the side condition of the rule is true. Between the application of BAL(L) that uses F and the goal $G_1 \dot{=} H_1$ in figure 10, there are no other applications of BAL(L), and G_1 is a bottom, $(F \cdot w(X_i))$, of the application of BAL(L). BAL(R) is now permitted provided it uses configuration G_i , $i \geq 1$, and the side condition holds. BAL(R) is not permitted using a configuration from a goal above $G_1 \dot{=} H_1$, even when the side condition is true. The strategy is to apply a

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

Figure 10. A potential switch from BAL(L) to BAL(R)

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

Figure 11. Part of a tableau

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

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

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

Fact 5 *If there are consecutive applications of BAL in a branch that use F and F' , then there is a word u such that $F' = (F \cdot u)$.*

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

Example 2 An initial part of the tableau, using the simple grammar of example 1 of section 2, is in figure 11. At goal (*), BAL(L) is applied. Either of the premises (1) and (2) could be the initial premise for the application: however, by the discussion above it is the lower premise (2). The first application of BAL(L) uses $F_0 = AA^5$ and the second uses $F_1 = AA^6$ and $F_1 = (F_0 \cdot ab)$. \square

Example 3 The initial part of the tableau for example 1, above is in figure 12. The

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

$$\frac{YYX + YZ + Z \doteq YYYX + YYZ + YZ + Z}{\text{BAL(L)}} \quad \text{where (1) is the subtableau}$$

$$\frac{\frac{(**) \quad X \doteq YX + Z}{\text{UNF}}}{\frac{X \doteq YYX + YZ + Z}{\text{BAL(R)}}} X \doteq X \quad \epsilon \doteq \epsilon$$

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

$$\frac{X \doteq YYYX + YYZ + YZ + Z}{\text{BAL(R)}} \quad X \doteq YYX + YZ + Z$$

Figure 12. Part of a tableau

Unsuccessful final goals

$$E \doteq F \quad (\text{exactly one of } E, F \text{ is } \emptyset)$$

Figure 13. Unsuccessful final goals

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

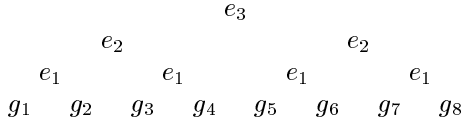
4.1. Successful tableaux

The missing ingredient in the tableau description is when a current goal is final. As in section 2, final goals are either *successful* or *unsuccessful*. The definition of unsuccessful final goal is the same as for simple grammars and is presented in figure 13. The definition of successful goal in figure 5 is an identity goal or a goal that is repeated (with at least one application of UNF between the two occurrences). The definition of success works for simple grammars in the presence of the rule CUT. However, without CUT, as example 2 illustrates, goals in a branch may continually increase their size.

Instead, success will be defined for the last element of a sequence of goals that have the same heads and are related by a finite family of tail extensions. A repeating goal is a special case where the extension is the identity (ϵ). So, the definition of *extension*, definition 7 of section 3, is enlarged to goals.

Definition 2 Assume $E = E_1H_1 + \dots + E_nH_n$, $F = F_1H_1 + \dots + F_nH_n$, $E' = E'_1G_1 + \dots + E'_mG_m$ and $F' = F'_1G_1 + \dots + F'_mG_m$. Assume goal h is $E \doteq F$ and goal g is $E' \doteq F'$. Goal h extends g by extension e , if E extends E' by e (and F extends F' by e).

The main combinatorial insight, “the extension theorem”, below, will underpin when a subgoal counts as a successful final goal. It involves families of extensions with the same heads. In this theorem, a goal $E \doteq F$ is true at level m , if $E \sim_m F$. To illustrate it, consider $n = 3$. Assume that there are goals $g(i), h(i), i : 1 \leq i \leq 8$ and each goal $g(i)$ has the form $E_1 G_1^i + \dots + E_3 G_3^i \doteq F_1 G_1^i + \dots + F_3 G_3^i$ and each goal $h(i)$ has the form $E_1 H_1^i + \dots + E_3 H_3^i \doteq F_1 H_1^i + \dots + F_3 H_3^i$. They all have the same heads. Assume extensions e_1, e_2 and e_3 as follows (where, for example, $g(2)$ extends $g(1)$ by e_1 and $g(5)$ extends $g(4)$ by e_3).



And assume the same extensions for the goals $h(i)$. The theorem says that if each $g(i), 1 \leq i \leq 8$, is true at level m and each $h(i), 1 \leq i < 8$, is true at level m , then $h(8)$ is also true at level m .

Proposition 1 [The extension theorem] *Assume there are two families of goals $g(i), h(i), 1 \leq i \leq 2^n$, and each goal $g(i)$ has the form $E_1 G_1^i + \dots + E_n G_n^i \doteq F_1 G_1^i + \dots + F_n G_n^i$ and each goal $h(i)$ has the form $E_1 H_1^i + \dots + E_n H_n^i \doteq F_1 H_1^i + \dots + F_n H_n^i$. Assume extensions e_1, \dots, e_n such that for each e_j and $i \geq 0$*

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

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

The proof of this result is inductive and is not presented here. A simple instance is now explained. Consider the tree of example 2. There is a branch where the goals are increasing in size: $YXA^5 \doteq CA^5, \dots, YXA^6 \doteq CA^6, \dots, YXA^7 \doteq CA^7, \dots$. And between these goals there is at least one application of UNF. To instantiate the extension theorem $n = 1$. The families of goals are: $g(1)$ is $YXG^1 \doteq CG^1$ where $G^1 = A^5$, $g(2) = h(1)$ which is $YXG^2 \doteq CG^2$ where $G^2 = A^6$ and $h(2)$ is $YXH^2 \doteq CH^2$ where $H^2 = A^7$. The extension is (A): $g(2)$ extends $g(1)$ by (A) and $h(2)$ extends $h(1)$ by (A). The theorem provides the following result: for any m , if $YXA^5 \sim_m CA^5$ and $YXA^6 \sim_m CA^6$, then $YXA^7 \sim_m CA^7$ which justifies that $YXA^7 \doteq CA^7$ is a successful final goal. The argument is similar to theorem 2 of section 2.5 for a repeat goal.

Definition 3 Assume a branch of goals $d(0), \dots, d(l)$. The goal $d(l), E_1 H_1 + \dots + E_n H_n \doteq F_1 H_1 + \dots + F_n H_n$, obeys the extension theorem if the following hold

1. There are families of goals $g(i), h(i), 1 \leq i \leq 2^n$ belonging to $\{d(0), \dots, d(l)\}$, and each goal $g(i)$ has the form $E_1 G_1^i + \dots + E_n G_n^i \doteq F_1 G_1^i + \dots + F_n G_n^i$ and each goal $h(i)$ has the form $E_1 H_1^i + \dots + E_n H_n^i \doteq F_1 H_1^i + \dots + F_n H_n^i$.
2. The goal $h(2^n)$ is $d(l)$ and there is at least one application of UNF between goal $h(2^n - 1)$ and $d(l)$.

Successful final goals

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

Figure 14. Successful final goals

3. There are extensions e_1, \dots, e_n such that for each e_j and $i \geq 0$

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

The second occurrence of a repeat goal in a branch obeys the extension theorem provided that there is at least one application of UNF between the two occurrences. Assume it has the form $E_1 G_1^i + \dots + E_n G_n^i \doteq F_1 G_1^i + \dots + F_n G_n^i$. Except for $h(2^n)$, the goals $g(i)$ and $h(i)$ are its first occurrence and each extension is the identity, (ϵ) . All three conditions of definition 2 are thereby satisfied.

The definition of successful final goal is given in figure 14. The following results are similar to those in section 2.

Theorem 1 *There is a unique boundedly finite tableau for goal $E \doteq F$.*

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

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

The bound on the size of the tableau for $E \doteq F$ is only primitive recursive with respect to the size of the DPDA (or, SDG) and, in particular, with respect to the number of states of the DPDA. Much more work is needed to check if this bound is anywhere near optimal.

References

- [1] Alur, R. and Madhusudan P. (2004). Visibly pushdown languages. *Procs. STOC 2004*, 202-211.
- [2] Baeten, J., Bergstra, J., and Klop, J. (1993). Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of ACM*, **40**, 653-682.
- [3] Büchi, J. (1962). On a decision method in restricted second order arithmetic. *Logic, Methodology and Philosophy of Science*, Proc. 1960 congress, Stanford Univ. Press, Stanford, CA, 1-11.
- [4] Burkart, O., Caucal, D., and Steffen, B. (1995). An elementary bisimulation decision procedure for arbitrary context-free processes. *Lecture Notes in Computer Science*, **969**, 423-433.
- [5] Burkart, O., and Steffen, B. (1994). Pushdown processes: parallel composition and model checking. *Lecture Notes in Computer Science*, **836**, 98-113.
- [6] Burkart, O., Caucal, D. Moller, F., and Steffen, B. (2001). Verification on infinite structures. In *Handbook of Process Algebra*, ed. Bergstra, J., Ponse, A., and Smolka, S., 545-623, North-Holland.
- [7] Caucal, D. (1992). On the regular structure of prefix rewriting. *Theoretical Computer Science*, **106**, 61-86.

- [8] Caucal, D. (1995). Bisimulation of context-free grammars and of pushdown automata. *CSLI Lecture Notes*, **53**, 85-106.
- [9] Caucal, D. (1996). On infinite transition graphs having a decidable monadic theory. *Lecture Notes in Computer Science*, **1099**, 194-205.
- [10] Christensen, S., Hirshfeld, Y. and Moller, F. (1993). Bisimulation equivalence is decidable for all basic parallel processes. *Lecture Notes in Computer Science*, **715**, 143-157.
- [11] Christensen, S., Hüttel, H., and Stirling, C. (1995). Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, **121**, 143-148.
- [12] Ginsberg, S., and Greibach, S. (1966). Deterministic context-free languages. *Information and Control*, 620-648.
- [13] Groote, J., and Hüttel, H. (1994). Undecidable equivalences for basic process algebra. *Information and Computation*, **115**, 354-371.
- [14] Harrison, M. (1978). *Introduction to Formal Language Theory*, Addison-Wesley.
- [15] Harrison, M., and Havel, I. (1973). Strict deterministic grammars. *Journal of Computer and System Sciences*, **7**, 237-277.
- [16] Harrison, M., Havel, I., and Yehudai, A. (1979). On equivalence of grammars through transformation trees. *Theoretical Computer Science*, **9**, 173-205.
- [17] Hopcroft, J., and Ullman, J. (1979). *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley.
- [18] Hüttel, H., and Stirling, C. (1991). Actions speak louder than words: proving bisimilarity for context free processes. *Proceedings 6th Annual Symposium on Logic in Computer Science*, IEEE Computer Science Press, 376-386.
- [19] Jancar, P. (1997). Decidability of bisimilarity for one-counter processes. *Lecture Notes in Computer Science*, **1256**, 549-559.
- [20] Korenjak, A and Hopcroft, J. (1966). Simple deterministic languages. *Procs. 7th Annual IEEE Symposium on Switching and Automata Theory*, 36-46.
- [21] Muller, D., and Schupp, P. (1985). The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, **37**, 51-75.
- [22] Oyamaguchi, M., Honda, N., and Inagaki, Y. (1980). The equivalence problem for real-time strict deterministic languages. *Information and Control*, **45**, 90-115.
- [23] Rabin, M. (1969). Decidability of second-order theories and automata on infinite trees. *Transactions of American Mathematical Society*, **141**, 1-35.
- [24] Sénizergues, G. (1997). The equivalence problem for deterministic pushdown automata is decidable. *Lecture Notes in Computer Science*, **1256**, 671-681.
- [25] Sénizergues, G. (1998). Decidability of bisimulation equivalence for equational graphs of finite out-degree. *Procs. IEEE 39th FOCS*, 120-129.
- [26] Sénizergues, G. (2001). $L(A) = L(B)$? decidability results from complete formal systems. *Theoretical Computer Science*, **251**, 1-166.
- [27] Sénizergues, G. (2002). $L(A) = L(B)$? a simplified decidability proof. *Theoretical Computer Science*, **281**, 555-608.
- [28] Srba, J. (2002). Undecidability of weak bisimilarity for pushdown processes. *Lecture Notes in Computer Science*, **2421**, 579-593.
- [29] Stirling, C. (2001). Decidability of DPDA equivalence. *Theoretical Computer Science*, **255**, 1-31.
- [30] Stirling, C. (2002) Deciding DPDA equivalence is primitive recursive. *Lecture Notes in Computer Science*, **2380**, 821-832.