Language Theory and Infinite Graphs

Colin Stirling
School of Informatics
University of Edinburgh

Main aim of course

- Prove one result:

  language equivalence is decidable for deterministic context-free languages

- Was a notorious open problem in language theory

- Its solution paves the way for more difficult open problems from the 1960s: last lecture

- New applications: algorithmics of game semantics (equivalence of open programs in idealised algol, Ong et als)

Pushdown automata

A pushdown automaton, PDA, consists of

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

Basic transition $pX \xrightarrow{a} q\alpha$ where p, $q \in$ P, $X \in$ S, $a \in$ A $\cup \{\epsilon\}$ and $\alpha \in$ S*

Configurations

- A configuration $p\beta$, $p \in \mathsf{P}$ and $\beta \in \mathsf{S}^*$
- Transitions of a configuration

  If $pX \xrightarrow{a} q\alpha \in \mathsf{T}$ then $pX\delta \xrightarrow{a} q\alpha\delta$

Disjointness assumption

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

Configurations are unstable, only have ϵ-transitions, or stable, have no ϵ-transitions

Transition graphs $\mathsf{G}(p\beta)$

Generated by deriving all possible transitions from $p\beta$ and every configuration reachable from it

Example

$\mathsf{P} = \{p\}$, $\mathsf{S} = \{X\}$, $\mathsf{A} = \{a\}$, $\mathsf{T} = \{pX \xrightarrow{a} pXX\}$

$\mathsf{G}(pX)$ is:

$$pX \xrightarrow{a} pXX \xrightarrow{a} pXXX \xrightarrow{a} pXXXX \xrightarrow{a} \ldots$$

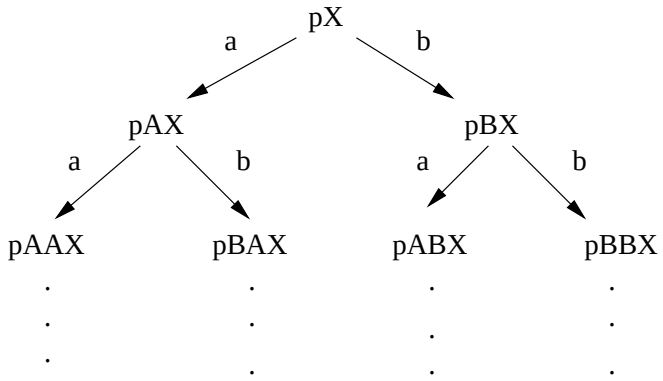
Transition $pXXX \xrightarrow{a} pXXXX$ follows from $pX \xrightarrow{a} pXX \in \mathsf{T}$

$$pX\ XX \xrightarrow{a} pXX\ XX$$

Example

P $= \{p\}$, S $= \{X, A, B\}$ and A $= \{a, b\}$, T is

$$pX \xrightarrow{a} pAX \qquad pA \xrightarrow{a} pAA \qquad pB \xrightarrow{a} pAB$$
$$pX \xrightarrow{b} pBX \qquad pA \xrightarrow{b} pBA \qquad pB \xrightarrow{b} pBB$$



Example

P $= \{p, q, r\}$, S $= \{X\}$, A $= \{a, b\}$ and T is

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

G(pX) is

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

Normal form (up to graph isomorphism)

- If $pX \xrightarrow{a} q\alpha \in \mathsf{T}$ then the length of α, $|\alpha|$, is at most 2.

Achieving normal form

- by introducing extra stack symbols
- assume maximum length of any α in $pX \xrightarrow{a} q\alpha \in \mathsf{T}$ is n
- new stack symbols $[\beta]$, $1 < |\beta| \leq n$ are introduced

Example

$$pX \xrightarrow{a} pX_1X_2X_3X_4 \qquad pX_1 \xrightarrow{a} p\epsilon \qquad pX_4 \xrightarrow{a} pX_1$$
$$pX_2 \xrightarrow{a} pX_1X_3X_4 \qquad pX_3 \xrightarrow{a} pX_1X_1X_1X_4$$

This set is transformed as follows when starting from pX

- $pX \xrightarrow{a} pX_1[X_2X_3X_4]$ and $p[X_2X_3X_4] \xrightarrow{a} p[X_1X_3X_4][X_3X_4]$
- $p[X_1X_3X_4] \xrightarrow{a} p[X_3X_4]$ and $p[X_3X_4] \xrightarrow{a} p[X_1X_1X_1X_4]X_4$
- $p[X_1X_1X_1X_4] \xrightarrow{a} p[X_1X_1X_4]$ and $p[X_1X_1X_4] \xrightarrow{a} p[X_1X_4]$
- $p[X_1X_4] \xrightarrow{a} pX_4$ and transitions for pX_1 and pX_4 are unchanged

Bounded in/out degree of $\mathsf{G}(p\alpha)$

- Out degree of terminal configuration $p\alpha$ is 0
- Out degree of $pX\alpha$ is bounded by the number of basic transitions in T of form $pX \xrightarrow{a} q\beta$
- In degree of $p\alpha$ is bounded by the number of basic transitions in T of form $qY \xrightarrow{a} p\alpha'$, where α' is a prefix of α

Word transitions

- extend transitions to words $w \in \mathsf{A}^*$
- $p\alpha \xrightarrow{w} q\beta$ if that there is a w-path from $p\alpha$ to $q\beta$ in $\mathsf{G}(p\alpha)$
- so, $p\alpha \xrightarrow{\epsilon} p\alpha$ for any $p\alpha$
- role of ϵ-transitions in T thereby differs from a-transitions in T when $a \in \mathsf{A}$, because ϵ is the empty word

Example

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

$pXX \xrightarrow{ab} r\epsilon$ because $pXX \xrightarrow{a} pXXX \xrightarrow{b} rXX \xrightarrow{\epsilon} rX \xrightarrow{\epsilon} r\epsilon$

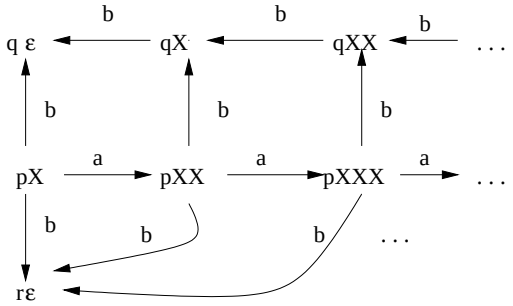
Collapsed transition graphs $\mathsf{G}^c(p\alpha)$

- consider word transitions $\xrightarrow{a}$, $a \in \mathsf{A}$, between stable configurations
- for $p\alpha$, $q\beta$ stable, $p\alpha \xrightarrow{a} q\beta$ if $p\alpha \xrightarrow{a} r\gamma(\xrightarrow{\epsilon})^* q\beta$
- this is the collapsed graph, without ϵ-transitions
- if $p\alpha$ is stable then $\mathsf{G}^c(p\alpha)$ is its collapsed graph
- $\mathsf{G}(p\alpha) = \mathsf{G}^c(p\alpha)$ if the PDA does not have ϵ-transitions

Example

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

So $\mathsf{G}^c(pX)$ is

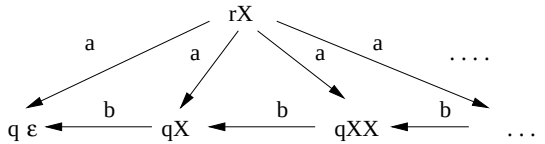


Infinite out degree

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

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

$G^c(rX)$ is



Infinite out degree only if there is nondeterminism in basic ϵ-transitions

Exercise

If $\mathsf{G}(pX)$ is

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

then what is $\mathsf{G}^{\mathsf{c}}(pX)$?

Subclasses of pushdown automata

- real-time: there are no ϵ-transitions
- single-state: 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 $a \in$ A and $pX \xrightarrow{a} q\beta$, $pX \xrightarrow{a} r\gamma \in$ T, then $q = r$ and $\beta = \gamma$
- deterministic: if it 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 $q\beta \xrightarrow{u} r\epsilon$

Normal form for ϵ-determinisitic PDA

- if $pX \xrightarrow{\epsilon} q\beta \in \mathsf{T}$ then $\beta = \epsilon$ ϵ-transitions only pop the stack
- Equivalence is isomorphism of collapsed graphs

To prove this, there are three cases to consider for each offending transition

$$pX \xrightarrow{\epsilon} p_1 X_1 \alpha_1$$

Proof of normal form

1. If $pX \xrightarrow{\epsilon} p_1X_1\alpha_1 \xrightarrow{\epsilon} p_2X_2\alpha_2 \xrightarrow{\epsilon} \ldots$ then there are no configurations $pX\beta$ in collapsed graph. Therefore, $pX \xrightarrow{\epsilon} p_1X_1\alpha_1$ and any transition $qY \xrightarrow{a} pX\alpha$, $a \in \mathsf{A} \cup \{\epsilon\}$, are removed from T.

2. If $pX \xrightarrow{\epsilon} p_1X_1\alpha_1 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} p_nX_n\alpha_n$ and $p_nX\alpha_n$ is stable then we remove $pX \xrightarrow{\epsilon} p_1X_1\alpha_1$ from T and for each transition $p_nX_n \xrightarrow{a} q\beta \in \mathsf{T}$ add the transition $pX \xrightarrow{a} q\beta\alpha_n$ to T.

3. If $pX \xrightarrow{\epsilon} p_1X_1\alpha_1 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} p_n\epsilon$ then we remove $pX \xrightarrow{\epsilon} p_1X_1\alpha_1$ from T and add the new transition $pX \xrightarrow{\epsilon} p_n\epsilon$ to T.

Decision questions I

- Model checking: $\mathsf{G}^c(q\beta)$ is a model with respect to logical formulae
- Example, monadic second-order logic: first-order logic (with equality and atomic predicates E_a for $a \in \mathsf{A}$ with interpretation $\xrightarrow{a}$) + quantification over sets of vertices.

  Given $\mathsf{G}^c(q\beta)$ and $\Phi(x)$ with one free variable, is $\Phi(x)$ true at $q\beta$?
- Long history here: Büchi 1962 (example slide 5), Rabin 1968 (example slide 6), Muller and Schupp 1985 (real-time PDA) and Caucal 1996 (all PDA = prefix-recognisable graphs)
- Standard temporal logics, LTL, CTL, CTL* and modal μ-calculus, are sublogics

Decision questions II

- Equivalence checking: given $p\alpha$ and $q\beta$ are they equivalent ?
- Classically, formal language theory views the language generable from a configuration as paramount.
- $L(p\alpha) = \{w \in \mathsf{A}^* : p\alpha \xrightarrow{w} q\epsilon \text{ for some } q \in \mathsf{P}\}$ Empty stack acceptance
- Languages accepted by PDA are the context-free languages

Example

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

$L(pX)$ is $\{a^n b^{n+1} : n \geq 0\} \cup \{a^n b : n \geq 0\}$.

Language equivalence

- In 1960s: $L(p\alpha) = L(q\beta)$ undecidable for single-state real-time PDA
- Active research question of 1960/70s: is $L(p\alpha) = L(q\beta)$ decidable for deterministic PDA? The DPDA equivalence problem.
  - Decidable for real-time single-state DPDA , Korenjak and Hopcroft in 1966. (Exponential algorithm. Improved to polynomial time.)
  - Decidable for single stack ...
  - Decidable for finite-turn ...
  - ⋮
  - Decidable for real-time DPDA, Oyamaguchi, Honda and Inagaki in 1980. (No complexity bound: procedure is two semi-decision procedures.)

Positive solution of DPDA problem

- Sénizergues (full proof, 166 pages, 2001). Configuration notation not rich enough. Deeper algebraic structure needed. Sénizergues's proof is primarily algebraic. Simplified proof (Stirling 2001) using bisimulation equivalence and graphs. Redone algebraically (Sénizergues 2002).

- Solution is two semi-decision procedures: one for inequivalence, find a smallest distinguishing word, one for equivalence, a nondeterministic proof procedure. No complexity upper bound.

- Simpler deterministic decision procedure with a primitive recursive complexity upper bound (Stirling 2002) using bisimulation equivalence and graphs $\mathsf{G}^c(p\alpha)$. Topic of these lectures.

Aside: final state vs empty stack

- A more classical definition of a PDA includes final states F $\subseteq$ P
- $L(p\alpha)$ is $\{w \in \mathsf{A}^* : p\alpha \xrightarrow{w} q\beta,\ q \in \mathsf{F}\}$ Final state acceptance
- For PDA, languages recognised under final state acceptance coincide with those recognised under empty stack acceptance
- Not true for DPDA. Languages recognised under final state acceptance are deterministic context-free languages. Languages accepted under empty stack acceptance is a proper subset. The DPDA equivalence problem is really about deterministic context-free languages.

Empty stack acceptance is OK

For any deterministic context-free language L, there is a DPDA configuration that accepts $L\$$ where $\$ \notin$ A is an end of word marker

$$L\$ = \{w\$ \ : \ w \in L\}$$

Proof

- Add new bottom of stack symbol B to a DPDA with final state acceptance: $p\alpha \xrightarrow{u} q\beta$ iff $p\alpha B \xrightarrow{u} q\beta B$ in amended DPDA. Construct DPDA with empty stack acceptance from amended DPDA.

- If for final q and X, $qX \xrightarrow{\epsilon} r\epsilon \in$ T and r not final, then add new final r' and replace transition with $qX \xrightarrow{\epsilon} r'\epsilon$ and add transitions for r' same as for r.

- For each final q and X, if qX is stable then add new transition $qX \xrightarrow{\$} e\epsilon$ where e erases all stack elements (including B), so $eY \xrightarrow{\varepsilon} e\varepsilon$ for all Y.

- So, $p\alpha B \xrightarrow{u} q\beta B$ and q is final in amended DPDA with final state acceptance iff $p\alpha B \xrightarrow{u\$} e\epsilon$ in DPDA that accepts with empty stack acceptance.

Bisimulation equivalence

- Based on existence of a binary relation between vertices of a transition graph which is preserved by transitions.
- R is a bisimulation on vertices 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 $(E', F') \in R$
- $E \sim F$ if there is a bisimulation containing (E, F)
- $\sim$ is bisimulation equivalence

Example

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

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

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

The proof that R is a bisimulation relation is straightforward

Example

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

$pX \not\sim rX$ although $L(pX) = L(rX)$.

The problem is how to match the transition $rX \xrightarrow{a} r_1X$.

Example

$$\begin{array}{ccccccccc} pX & \xrightarrow{a} & pXX & \xrightarrow{a} & pXXX & \xrightarrow{a} & pXXXX & \xrightarrow{a} & \ldots \\ qY & \xrightarrow{a} & q_1Y & \xrightarrow{a} & qYY & \xrightarrow{a} & q_1YY & \xrightarrow{a} & \ldots \end{array}$$

The bisimulation R which witnesses $pX \sim qY$ is infinite

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

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

Bisimulation approximants

The family $\{\sim_n : n \geq 0\}$ is defined inductively

$E \sim_0 F$ for all vertices E, F

$E \sim_{n+1} F$ iff for all $a \in \mathsf{A}$

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

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

Decision question $p\alpha \sim q\beta$?

- Decidable for normed single-state real-time PDA, irredundant context-free grammars. (Baeten, Bergstra and Klop 1987). Extended to all single-state real-time PDA, all context-free grammars. (Christensen, Hüttel and Stirling 1992).

- Decidability extended to normed real-time PDA, real-time PDA and collapsed graphs of ϵ-deterministic PDA. (Stirling 1996, Sènizergues 2002)

- Undecidable for arbitrary collapsed graphs of PDA (a consequence of Srba 2002).

Language and bisimulation equivalence

- Fact For any collapsed graph of a PDA, if $q\beta \sim r\delta$ then $L(q\beta) = L(r\delta)$

- Converse only holds if the PDA is deterministic and normed. Proof: verify then that the following relation on collapsed graph is a bisimulation.

$$\{(q\beta, r\delta) \ : \ L(q\beta) = L(r\delta)\}$$

Language Theory and Infinite Graphs

Colin Stirling
School of Informatics
University of Edinburgh

Example

P $= \{p, q, r\}$, S $= \{X\}$, A $= \{a, b\}$ and T is

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

G(pX) is

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

Pushdown grammars

Real-time single-state PDA

A pushdown grammar, PDG, consists of

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

Basic transition $X \xrightarrow{a} \alpha$ where $X \in$ S, $a \in$ A and $\alpha \in$ S*

Configurations

- A configuration, $\beta \in \mathsf{S}^*$
- Transitions of a configuration

  If $X \xrightarrow{a} \alpha \in \mathsf{T}$ then $X\delta \xrightarrow{a} \alpha\delta$

Normal form

Assume that a PDG is in normal form

- If $X \xrightarrow{a} \alpha \in \mathsf{T}$ then $|\alpha| \leq 2$ (To ensure this, add extra stack symbols.)
- Normed: for every $X \in \mathsf{S}$ there is a word u such that $u \in L(X)$

Some definitions

Assume a fixed total ordering on A

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

For 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 $|w(X)|$

Ensuring normedness

Can compute $w(X)$, if it exists

- Identify stack symbols with norm 1: $X \xrightarrow{a} \epsilon \in \mathsf{T}$
- Identify stack symbols with norm 2: $X \xrightarrow{a} Z \in \mathsf{T}$ and Z has norm 1
- Identify stack symbols with norm 3: $X \xrightarrow{a} Z \in \mathsf{T}$ and Z has norm 2 or $X \xrightarrow{a} YZ \in \mathsf{T}$ and both Y and Z have norm 1
- ...
- 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 $2k+1$

Normedness continued

If X is 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

Result a normed PDG that preserves language equivalence (for configurations α such that $L(\alpha) \neq \emptyset$)

Maximum norm M of a PDG: $\max\{|w(X)| : X \in \mathsf{S}\}$

M can be exponential in the number of stack symbols

Norm extends to configurations: $w(\alpha)$ is the unique smallest word v such that $\alpha \xrightarrow{v} \epsilon$. Norm of α is $|w(\alpha)|$.

Simple Grammars

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 similar method to that presented here.

A simple grammar is a deterministic PDG in normal form

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

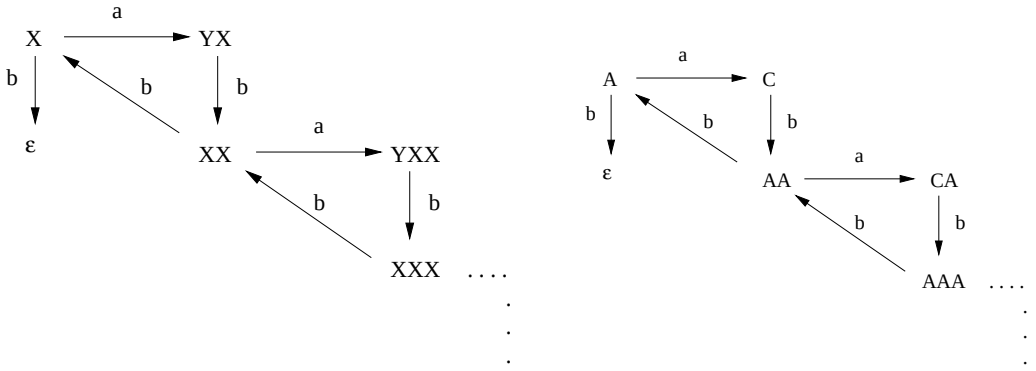
Recognition power of simple grammars is less than that of DPDA:

$L = \{a^n b^n \ : \ n > 0\} \cup \{a^n c \ : \ n > 0\}$ is generable by a DPDA but not by a simple grammar

Example

$X \xrightarrow{a} YX \quad X \xrightarrow{b} \epsilon \quad Y \xrightarrow{b} X \qquad A \xrightarrow{a} C \quad A \xrightarrow{b} \epsilon \quad C \xrightarrow{b} AA$

G(X) and G(A) are



$w(X) = b$ and $w(A) = b$ and $w(Y) = bb$ and $w(C) = bbb$

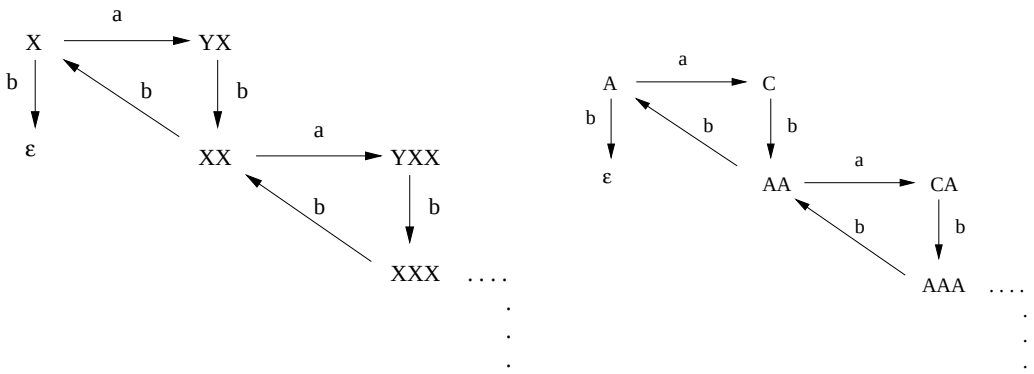
Maximum norm $\mathrm{M} = 3$

Notation

- Add an extra configuration $\emptyset$ which is unnormed, $|\emptyset|$, is 0 and $L(\emptyset) = \emptyset$
- $\alpha \cdot u$ is either the configuration β such that $\alpha \xrightarrow{u} \beta$ or it is the deadlocked configuration $\emptyset$

  So, for every α and u, $\alpha \cdot u$ is unique

Example



$(YX \cdot bab) = XXX$

$(AA \cdot aa) = \emptyset$

Decision Problem

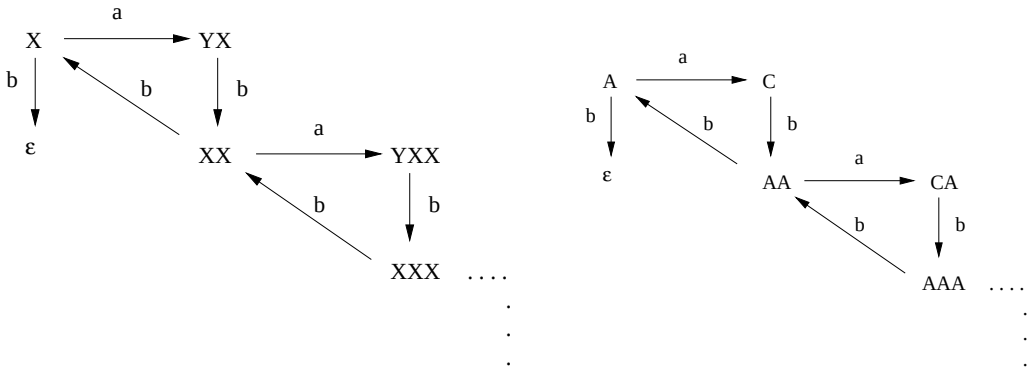
Given $\alpha, \beta \in \mathsf{S}^*$, is $L(\alpha) = L(\beta)$?

The problem $L(\alpha) \subseteq L(\beta)$? is undecidable

Because simple grammars are deterministic and normed language equivalence coincides with bismulation equivalence

$L(\alpha) = L(\beta)$ iff $\alpha \sim \beta$

Example



$A \sim X$. The bisimulation justifying equivalence is infinite

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

Decision Procedure

- Decision procedure 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 $\alpha \sim \beta$?

Rules: UNF (unfold)

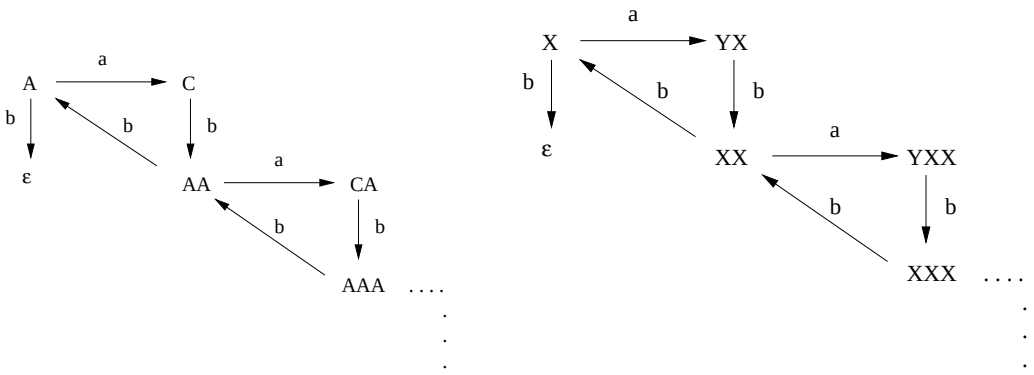
Goal, $\alpha \doteq \beta$ reduces to subgoals $(\alpha \cdot a) \doteq (\beta \cdot a)$ for each $a \in \mathsf{A}$

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

Fact If $\alpha \sim \beta$, then for all $a \in \mathsf{A}$, $(\alpha \cdot a) \sim (\beta \cdot a)$

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

Example



$$\dfrac{\dfrac{\dfrac{A \dot{=} X}{C \dot{=} YX}\,\text{UNF} \qquad \epsilon \dot{=} \epsilon}{\emptyset \dot{=} \emptyset \qquad AA \dot{=} XX}\,\text{UNF}}{CA \dot{=} YXX \quad A \dot{=} X}\,\text{UNF}$$

Imbalance

- Imbalance of a goal $\alpha \doteq \beta$ 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 between α and β is $\max\{|\alpha_1|, |\beta_1|\}$
- If the imbalance between α and β is 0, then $\alpha = \beta$, so $\alpha \sim \beta$

Rules: BAL(L) and BAL(R) (balance)

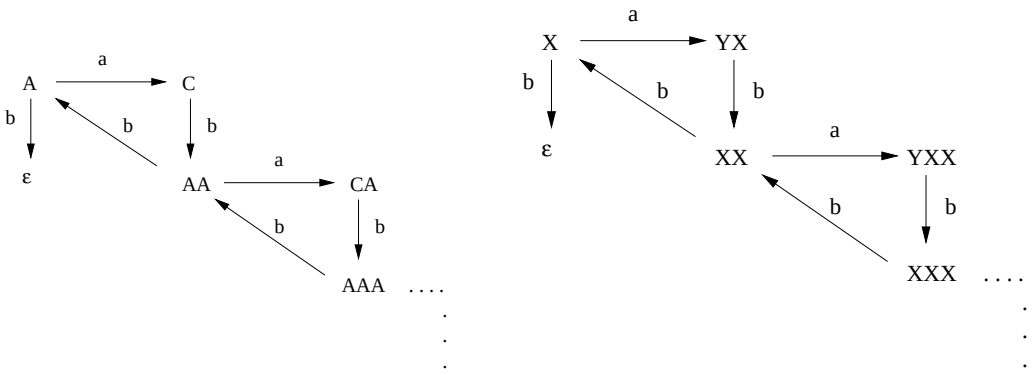
$$\frac{\begin{array}{rcl} X\alpha & \dot{=} & \beta \\ & \vdots & \\ \alpha'\alpha & \dot{=} & \beta' \end{array}}{\alpha'(\beta \cdot w(X)) \dot{=} \beta'}\ \mathsf{C} \qquad \frac{\begin{array}{rcl} \beta & \dot{=} & X\alpha \\ & \vdots & \\ \beta' & \dot{=} & \alpha'\alpha \end{array}}{\beta' \dot{=} \alpha'(\beta \cdot w(X))}\ \mathsf{C}$$

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

Example



$$\dfrac{\dfrac{AA^{220} \doteq X^{221}}{CA^{220} \doteq YX^{221}}\ \text{UNF}}{C(X^{221} \cdot w(A)) \doteq YX^{221}}\ \text{BAL} \quad \cdots$$

$w(A) = b$ and so last goal is balanced: $C(X^{221} \cdot b)$ is CX^{220}

BAL continued

Fact

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

2. If $\alpha \sim \beta$ then $\delta\alpha \sim \delta\beta$

Fact

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

2. If $\alpha \sim_n \beta$ and $|\delta| \geq k$ then $\delta\alpha \sim_{n+k} \delta\beta$

Rules: CUT

The final rule CUT allows common tails to be cut from a goal:

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

Example

$$\frac{CX^{220} \doteq YX^{221}}{C \doteq YX}\ \mathrm{CUT}$$

CUT cont

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

Fact If $\alpha\delta \not\sim_n \beta\delta$ then $\alpha \not\sim_n \beta$

Final goals

Successful final goals

$$\alpha \dot{=} \alpha \qquad \begin{array}{l} \alpha \dot{=} \beta \\ \vdots \quad \text{UNF at least once} \\ \alpha \dot{=} \beta \end{array}$$

Unsuccessful final goals

$\alpha \dot{=} \beta$ (exactly one of α, β is $\emptyset$)

Procedure

- Start with an initial goal, $\alpha \doteq \beta$, and deterministically build a proof tree by applying the tableau rules (there is an ordering on the rules)

- Goals are thereby reduced to subgoals. Rules are not applied to final goals

A successful tableau is a finite proof tree all of whose leaves are successful final goals

Otherwise a tableau is unsuccessful

Rule order

We assume that a BAL rule applies to a goal using the premise above which is the closest

Order on applying tableau rules

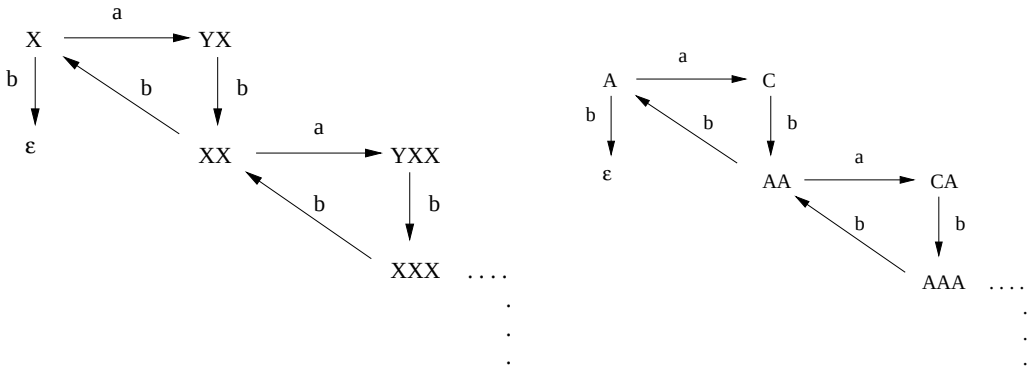
1. Apply BAL(L) followed immediately, if applicable, by CUT, or

2. Apply BAL(R) followed immediately, if applicable, by CUT, or

3. Apply UNF

Fact Any goal $\alpha \doteq \beta$ has a unique tableau

Example



$$\dfrac{\dfrac{\dfrac{\dfrac{A \dot{=} X}{\dfrac{C \dot{=} YX}{AA \dot{=} XX}\,\text{UNF}}\,\text{UNF}}{CA \dot{=} YXX}\,\text{BAL(L)}}{CX \dot{=} YXX}\,\text{CUT}}{C \dot{=} YX} \qquad \dfrac{\epsilon \dot{=} \epsilon}{A \dot{=} X}\,\text{UNF}$$

Decidability

Propositions

- Every tableau is finite
- $\alpha \sim \beta$ iff the tableau with root $\alpha \doteq \beta$ is successful

Main point, given $\alpha \doteq \beta$, then every goal in the tableau has a bounded size (in terms of $|\alpha|, |\beta|$ and the size of the PDG)

(Better procedure uses unique prime decomposition: leads to a polynomial time procedure.)

Comments

What are key features that underpin decidability?

- Bisimulation equivalence is a congruence with respect to stack prefixing

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

  Congruence allows us to tear apart a configuration $\alpha'\alpha$ and replace its tail with a potentially equivalent configuration β, $\alpha'\beta$

- Bisimulation equivalence supports cancellation of postfixed stacks

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

  Cancellation, as used in CUT, has the consequence that goals become small

Back to DPDA

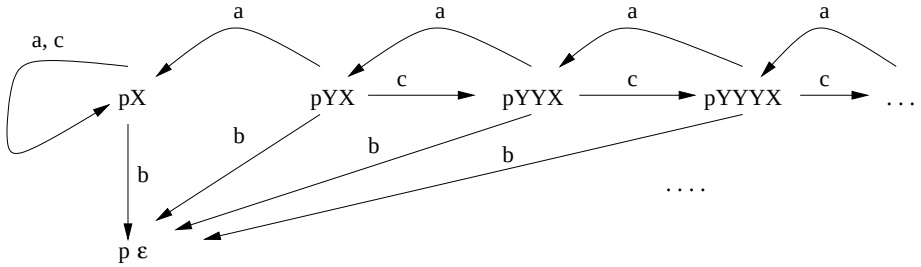
- How can we tear apart DPDA (stable) configurations $p\alpha$ and replace parts with parts? What is a part of a configuration?

  A configuration has state as well as stack

- $L(p\alpha) = L(q\beta)$ does not, in general, imply $L(p\delta\alpha) = L(q\delta\beta)$

- $L(p\alpha\delta) = L(q\beta\delta)$ does not, in general, imply $L(p\alpha) = L(q\beta)$

Even worse



- A main attack on the decision problem in the 1960/70s examined differences between stack lengths and potentially equivalent configurations that eventually resulted in a proof of decidability for real-time DPDAs

- However, $L(pY^nX) = L(pY^mX)$ for every m and n

However

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 postfixed cancellation

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

2. $L(\alpha) = L(\beta)$ iff $L(\delta\alpha) = L(\delta\beta)$

3. $\alpha \sim \beta$ iff $\alpha\delta \sim \beta\delta$

4. $\alpha \sim \beta$ iff $\alpha\delta \sim \beta\delta$

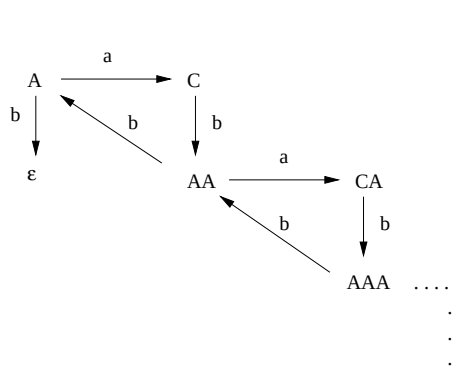
Therefore ...

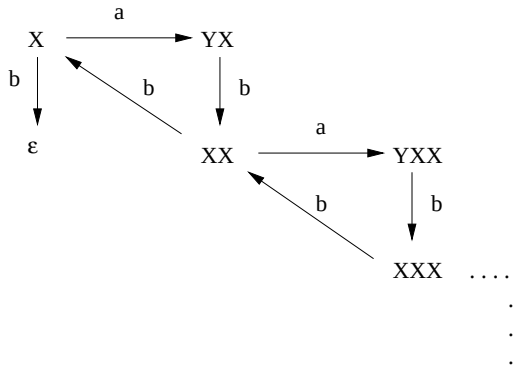
- This suggests that we should try to remain with PDGs
- Deterministic PDGs are too restricted
- Arbitrary PDGs are too rich (as they generate all the non-empty context-free languages)
- What is needed is a mechanism for constraining nondeterminism in a PDG

Language Theory and Infinite Graphs

Colin Stirling
School of Informatics
University of Edinburgh

Example





$$\dfrac{\dfrac{\dfrac{\dfrac{A \doteq X}{C \doteq YX}\,\text{UNF}}{AA \doteq XX}\,\text{UNF} \qquad \dfrac{A \doteq X}{\epsilon \doteq \epsilon}\,\text{UNF}}{CA \doteq YXX}\,\text{BAL(L)}}{\dfrac{CX \doteq YXX}{C \doteq YX}\,\text{CUT}}$$

Decidability

Sketched decidability proof of language equivalence for simple grammars using tableau proof system

Propositions

- Every tableau is finite
- $\alpha \sim \beta$ iff the tableau with root $\alpha \doteq \beta$ is successful

Main point, given $\alpha \doteq \beta$, then every goal in the tableau has a bounded size (in terms of $|\alpha|, |\beta|$ and the size of the PDG)

Comments

What are key features that underpin decidability?

- Bisimulation equivalence is a congruence with respect to stack prefixing

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

  Congruence allows us to tear apart a configuration $\alpha'\alpha$ and replace its tail with a potentially equivalent configuration β, $\alpha'\beta$

- Bisimulation equivalence supports cancellation of postfixed stacks

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

  Cancellation, as used in CUT, has the consequence that goals become small

Back to DPDA

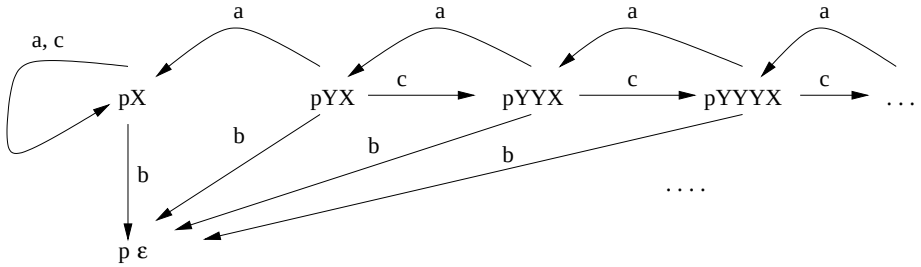
- How can we tear apart DPDA (stable) configurations $p\alpha$ and replace parts with parts? What is a part of a configuration?

  A configuration has state as well as stack

- $L(p\alpha) = L(q\beta)$ does not, in general, imply $L(p\delta\alpha) = L(q\delta\beta)$

- $L(p\alpha\delta) = L(q\beta\delta)$ does not, in general, imply $L(p\alpha) = L(q\beta)$

Even worse



- A main attack on the decision problem in the 1960/70s examined differences between stack lengths and potentially equivalent configurations that eventually resulted in a proof of decidability for real-time DPDAs

- However, $L(pY^nX) = L(pY^mX)$ for every m and n

However

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 postfixed cancellation

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

2. $L(\alpha) = L(\beta)$ iff $L(\delta\alpha) = L(\delta\beta)$

3. $\alpha \sim \beta$ iff $\alpha\delta \sim \beta\delta$

4. $\alpha \sim \beta$ iff $\alpha\delta \sim \beta\delta$

Therefore ...

- This suggests that we should try to remain with PDGs
- Deterministic PDGs are too restricted
- 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

Textbook transformation of PDAs into PDGs

- From PDA to language equivalent PDGs (with ϵ-transitions)
- Introduce stack symbols $[pXq]$ so that $L([pXq]) = \{w : pX \xrightarrow{w} q\epsilon\}$
- Convert basic transitions into sets of transitions:

  $pX \xrightarrow{a} q\epsilon$ is $\{[pXq] \xrightarrow{a} \epsilon\}$

  $pX \xrightarrow{a} rY$ is $\{[pXq] \xrightarrow{a} [rYq]\}$

  $pX \xrightarrow{a} rYZ$ is $\{[pXq] \xrightarrow{a} [rYp'][p'Zq] : p' \in \mathsf{P}\}$

- $w \in L(pX_1 \dots X_n)$ iff $w \in L([pX_1q_1][q_1X_2q_2] \dots [q_{n-1}X_nq_n])$ for some q_i

Example

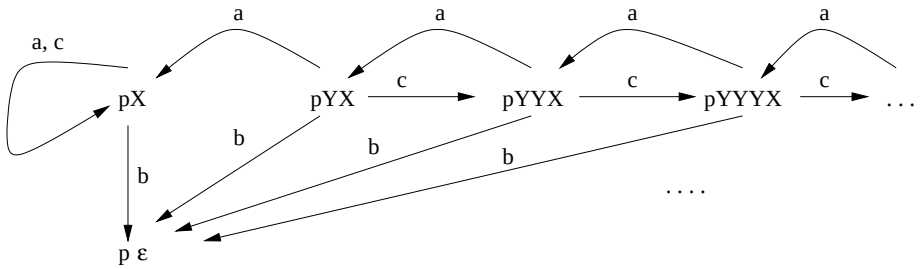
$$\begin{array}{ccccccc}
q\epsilon & \xleftarrow{b} & qX & \xleftarrow{b} & qXX & \xleftarrow{b} & \dots \\
\uparrow b & & \uparrow b & & \uparrow b & & \\
pX & \xrightarrow{a} & pXX & \xrightarrow{a} & pXXX & \xrightarrow{a} & \dots \\
\downarrow b & & \downarrow b & & \downarrow b & & \\
r\epsilon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

- $[pXq] \xrightarrow{a} [pXp][pXq] \dots$
- $[pXq] \xrightarrow{a} [pXq][qXq] \dots$
- Introducing extra nondeterminism

For DPDA

- Transform into normal form PDG without ϵ-transitions
- We only convert transitions for $a \in$ A
- A stack symbol $[pXq]$ is an ε-symbol, if $pX \xrightarrow{\epsilon} q\epsilon \in$ T. All ε-symbols are erased from the right hand side of any transition in the SDG
- Next, the SDG is normalised by removing all unnormed stack symbols

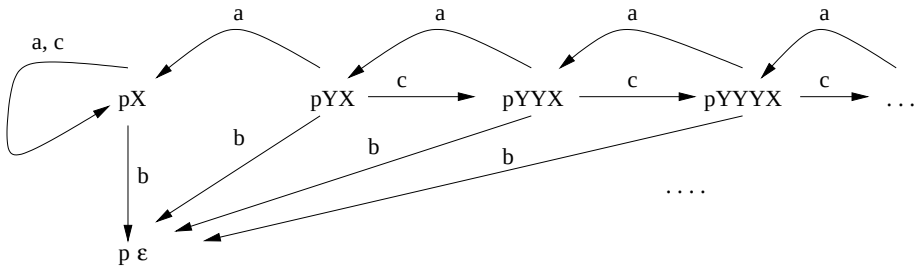
Example



$$[pXp] \xrightarrow{a} [pXp] \qquad [pXr] \xrightarrow{a} [pXr] \qquad [pXp] \xrightarrow{b} \epsilon$$
$$[pXp] \xrightarrow{c} [pXp] \qquad [pXr] \xrightarrow{c} [pXr]$$
$$[pYp] \xrightarrow{a} \epsilon \qquad [pYr] \xrightarrow{b} \epsilon \qquad [pYp] \xrightarrow{c} [pYp][pYp]$$
$$[pYp] \xrightarrow{c} [pYr][rYp] \qquad [pYr] \xrightarrow{c} [pYp][pYr] \qquad [pYr] \xrightarrow{c} [pYr][rYr]$$

There are two ϵ-stack symbols, $[rXp]$ and $[rYr]$

Example



$$[pXp] \xrightarrow{a} [pXp] \qquad [pXp] \xrightarrow{b} \epsilon \qquad [pXp] \xrightarrow{c} [pXp]$$
$$[pYp] \xrightarrow{a} \epsilon \qquad [pYr] \xrightarrow{b} \epsilon \qquad [pYp] \xrightarrow{c} [pYp][pYp]$$
$$[pYr] \xrightarrow{c} [pYp][pYr] \qquad [pYr] \xrightarrow{c} [pYr]$$

Extra nondeterminism: consider G($[pYr]$)

Main problem

- Transformation does not preserve bisimulation equivalence
- How to overcome this ?
- Need to preserve structure

Sum configurations

- Convert transitions into transitions over SUMS:

  $pX \xrightarrow{a} q\epsilon$ is $[pXq] \xrightarrow{a} \epsilon$

  $pX \xrightarrow{a} rY$ is $[pXq] \xrightarrow{a} [rYq]$

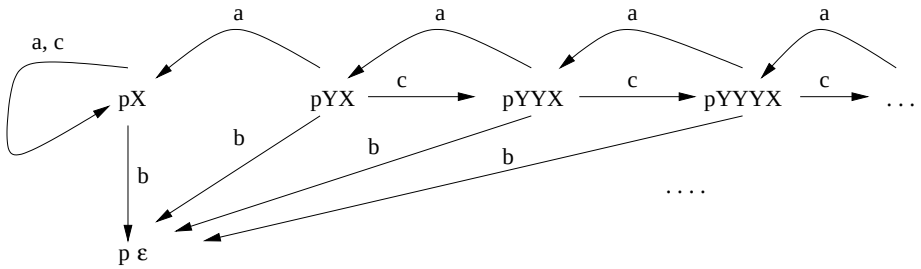
  $pX \xrightarrow{a} rYZ$ is $[pXq] \xrightarrow{a} \sum\{[rYp'][p'Zq] : p' \in \mathsf{P}\}$

- Convert $pX_1 \ldots X_n$ to $\sum\{[pX_1q_1][q_1X_2q_2] \ldots [q_{n-1}X_nq_n] : q_i \in \mathsf{P}\}$

- Language of a sum is union of languages of summands

- For DPDA determinize transitions for sum configurations: preserve structure

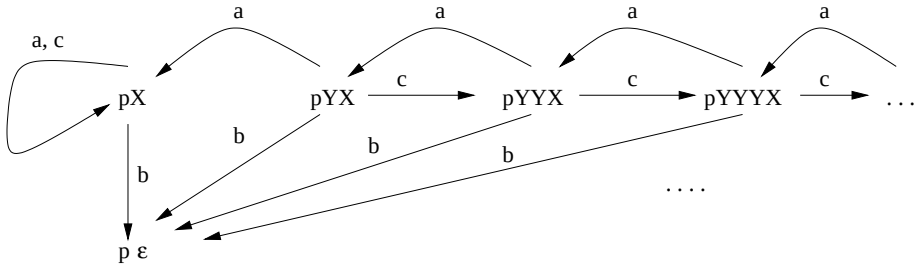
Example



$$[pXp] \xrightarrow{a} [pXp] \qquad [pXp] \xrightarrow{b} \epsilon \qquad [pXp] \xrightarrow{c} [pXp]$$
$$[pYp] \xrightarrow{a} \epsilon \qquad [pYr] \xrightarrow{b} \epsilon \qquad [pYp] \xrightarrow{c} [pYp][pYp]$$
$$[pYr] \xrightarrow{c} [pYp][pYr] \qquad [pYr] \xrightarrow{c} [pYr]$$

- Let $A = [pXp]$, $B = [pYp]$, $C = [pYr]$. So, pYX is $BA + C$

Determinized transitions



$$A \xrightarrow{a} A \qquad A \xrightarrow{b} \epsilon \quad A \xrightarrow{c} A$$

$$B \xrightarrow{a} \epsilon \qquad C \xrightarrow{b} \epsilon \quad B \xrightarrow{c} BB$$

$$C \xrightarrow{c} BC + C$$

- Let $A = [pXp]$, $B = [pYp]$, $C = [pYr]$. So, pYX is $BA + C$

Strict deterministic grammars

Due to Harrison and Havel in 1970s

Add an extra component to a PDG

- An equivalence relation $\equiv$ on S (stack symbols)

$\equiv$ partitions the stack symbols S into disjoint subsets $\mathsf{S}_1, \ldots, \mathsf{S}_k$:

for each i, and pair of stack symbols $X, Y \in \mathsf{S}_i$, $X \equiv Y$

INTUITION FROM DPDA: $X \equiv Y$ iff $X = [pZq]$ and $Y = [pZr]$

Sequences

Extend $\equiv$ on S to a relation on S*

$\alpha \equiv \beta$ iff

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

For instance, if $Y \equiv Z$ then $XY\alpha \equiv XZ$ for any α

$\equiv$ on stacks is not an equivalence relation

Properties

- $\alpha\beta \equiv \alpha$ 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 $\alpha \equiv \beta$

Strict deterministic

$\equiv$ on S is strict when

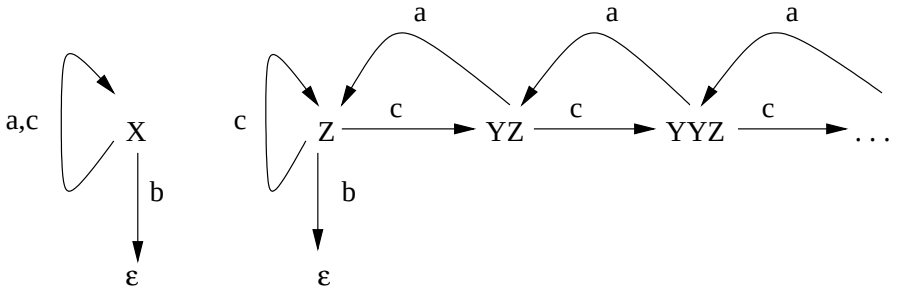
$$
\begin{array}{lcccc}
 & & X \xrightarrow{a} \alpha \in \mathsf{T} & & \alpha \\
1. & \text{if} & \| & \text{then} & \| \\
 & & Y \xrightarrow{a} \beta \in \mathsf{T} & & \beta
\end{array}
$$

$$
\begin{array}{lcccc}
 & & X \xrightarrow{a} \alpha \in \mathsf{T} & & X \\
2. & \text{if} & \| & \text{then} & \| \\
 & & Y \xrightarrow{a} \alpha \in \mathsf{T} & & Y
\end{array}
$$

A PDG with $\equiv$ is strict deterministic, an SDG, if its relation $\equiv$ is strict

Example

$$X \xrightarrow{a} X \qquad X \xrightarrow{b} \epsilon \qquad X \xrightarrow{c} X \qquad Y \xrightarrow{a} \epsilon$$
$$Y \xrightarrow{c} YY \qquad Z \xrightarrow{b} \epsilon \qquad Z \xrightarrow{c} Z \qquad Z \xrightarrow{c} YZ$$



Strict: partition $\{\{X\},\{Y,Z\}\}$

$Y \xrightarrow{c} YY$, $Z \xrightarrow{c} Z$, $Z \xrightarrow{c} YZ$ and $YY \equiv Z$, $YY \equiv YZ$ and $Z \equiv YZ$

Constrained nondeterminism

- If each set in the partition is a singleton then the SDG is deterministic, a simple grammar, and $\alpha \equiv \beta$ iff $\alpha = \beta$

- In general, constrained nondeterminism

  $X \xrightarrow{a} \epsilon \in \mathsf{T}$, $X \equiv Y$ and $X \neq Y$ implies no a-transition $Y \xrightarrow{a} \beta \in \mathsf{T}$

  Suppose $Y \xrightarrow{a} \beta$. By condition 1 of being strict $\epsilon \equiv \beta$, so $\beta = \epsilon$.

  By condition 2 of being strict $X = Y$, which is a contradiction.

Key property

Strictness conditions generalise to words and stacks

$$\text{1. if } \begin{array}{c} \alpha \xrightarrow{w} \alpha' \\ \| \\ \beta \xrightarrow{w} \beta' \end{array} \text{ then } \begin{array}{c} \alpha' \\ \| \\ \beta' \end{array}$$

$$\text{2. if } \begin{array}{c} \alpha \xrightarrow{w} \alpha' \\ \| \\ \beta \xrightarrow{a} \alpha' \end{array} \text{ then } \begin{array}{c} \alpha \\ \| \\ \beta \end{array}$$

Proof in the notes page 25

Corollaries

- If $\alpha \equiv \beta$ and $w \in L(\alpha)$, then for all words v, and $a \in$ A, $wav \notin L(\beta)$
- If $\alpha \equiv \beta$ and $\alpha \neq \beta$ then $L(\alpha) \cap L(\beta) = \emptyset$

DPDA intuition: properties hold for $[pXq]$, $[pXr]$

Sum configurations

- Extend configuration to sets of sequences of stack symbols, $\{\alpha_1, \ldots, \alpha_n\}$
- Write it as a sum $\alpha_1 + \ldots + \alpha_n$ without repeated elements
- Degenerate case is the empty sum, $\emptyset$
- $L(\alpha_1 + \ldots + \alpha_n) = \bigcup \{L(\alpha_i) : 1 \leq i \leq n\}$
- $L(\emptyset) = \emptyset$

Admissible sums

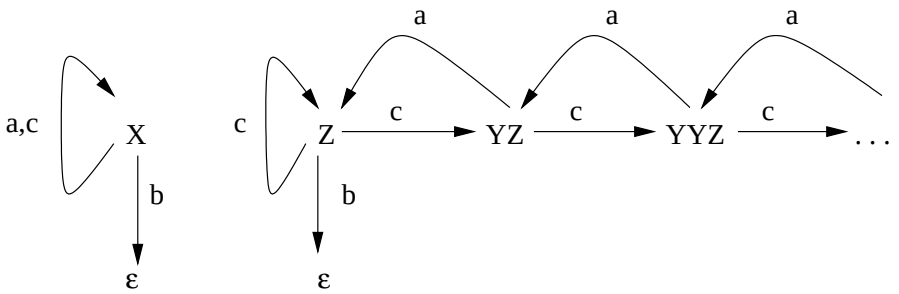
- $\beta_1 + \ldots + \beta_n$ is admissible if $\beta_i \equiv \beta_j$ for each i and j
- Admissible configurations are preserved by transitions

  If $\alpha_1 + \ldots + \alpha_n$ admissible then for all w the sum configuration

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

  is admissible
- DPDA intuition: $\{[pX_1q_1] \ldots [q_{n-1}X_nq_n] \,:\, q_i \in \mathrm{P}\}$ is admissible

Example



Strict: partition $\{\{X\},\{Y,Z\}\}$

Admissible: XX, $ZZZ + ZZY$, $YX + Z$, $Z + YZ$, $Z + YZ + YYZ$

Not admissible $X + Z$, $Y + \epsilon$

Determinization

- T is changed to T^{d}. For each stack symbol X and $a \in \mathsf{A}$, the family of transitions $X \xrightarrow{a} \alpha_1, \ldots, X \xrightarrow{a} \alpha_n \in \mathsf{T}$ is determinized

$$X \xrightarrow{a} \alpha_1 + \ldots + \alpha_n \in \mathsf{T}^{\mathsf{d}}$$

- For each X, a a unique transition $X \xrightarrow{a} \sum \alpha_i \in \mathsf{T}^{\mathsf{d}}$ as sum could be $\emptyset$

Determinization cont.

- Prefix rule for generating transitions is generalised

$$\text{If } X_i \xrightarrow{a} \alpha_1^i + \ldots + \alpha_{n_i}^i \quad \text{for each } i : 1 \leq i \leq m \text{ then}$$

$$X_1\beta_1 + \ldots + X_m\beta_m \xrightarrow{a} \begin{array}{ll} \alpha_1^1\beta_1 + \ldots + \alpha_{n_1}^1\beta_1 & + \\ \ldots \qquad\qquad \ldots & + \\ \alpha_1^m\beta_m + \ldots + \alpha_{n_m}^m\beta_m & \end{array}$$

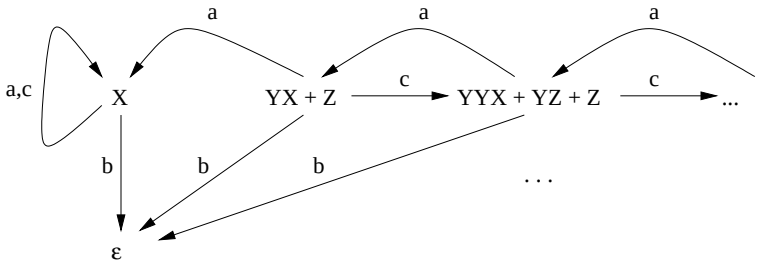
- Determinized graph $\mathsf{G}^d(\alpha_1 + \ldots + \alpha_n)$ using T^{d} and the extended prefix rule

Example

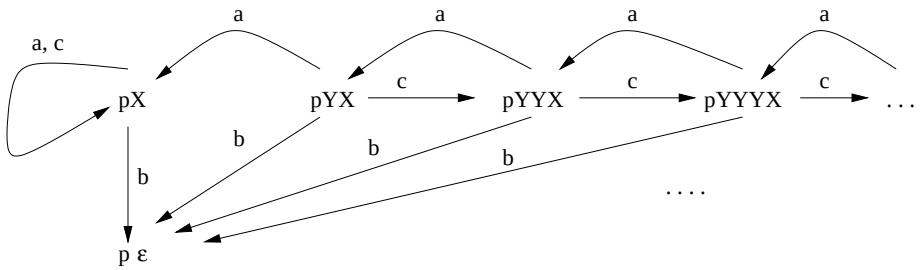
S $= \{X, Y, Z\}$, A $= \{a, b, c\}$ and the partition of S $= \{\{X\}, \{Y, Z\}\}$. And T^{d} is

$$X \xrightarrow{a} X \quad Y \xrightarrow{a} \epsilon \quad Z \xrightarrow{a} \emptyset \quad X \xrightarrow{b} \epsilon \quad Y \xrightarrow{b} \emptyset$$
$$Z \xrightarrow{b} \epsilon \quad X \xrightarrow{c} X \quad Y \xrightarrow{c} YY \quad Z \xrightarrow{c} YZ + Z$$

$\mathsf{G}^{\mathsf{d}}(YX + Z)$ is



Which is $\sim$



Characterising DPDA

- A DPDA can be converted into a determinized SDG, and configuration $p\alpha$ is converted into sum configuration $\mathrm{sum}(p\alpha)$ of the SDG where

  $\mathsf{G}^{\mathsf{c}}(p\alpha) \sim \mathsf{G}^{\mathsf{d}}(\mathrm{sum}(p\alpha))$

- (Conversely, a determinized SDG can be transformed into a DPDA)

- DPDA equivalence problem $=$ to deciding whether two admissible configurations of a determinized SDG are bisimulation equivalent

Notation

- E, F, G, ... range over admissible configurations
- $+$ 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 $E \cup F$
- Also use sequential composition, if E and F are admissible, then EF is the admissible configuration $\{\beta\gamma : \beta \in E \text{ and } \gamma \in F\}$
- $E = E_1G_1 + \ldots + E_nG_n$ is a head/tail form, if the head $E_1 + \ldots + E_n$ is admissible and at least one $E_i \neq \emptyset$, and each tail $G_i \neq \emptyset$

  If we write $E = E_1G_1 + \ldots + E_nG_n$ then E is a head/tail form

Congruence

- Decision question $E \sim F$?
- Assume $E = E_1 G_1 + \ldots + E_n G_n$
  - 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 $E \sim_{m+1} E_1 H_1 + \ldots + E_n H_n$
  - If each $H_i \neq \emptyset$ and for each j such that $E_j \neq \emptyset$, $H_j \sim G_j$, then $E \sim E_1 H_1 + \ldots + E_n H_n$
- We can tear apart configurations and replace parts with equivalent parts

Notation

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

Decision procedure

- Given by a deterministic tableau proof system where goals are reduced to subgoals
- Goals and subgoals have the form $E \doteq F$, is $E \sim F$
- Rules are (generalisations of) unfold, UNF, and balance rules,

  BAL(L) and BAL(R). (CUT free)

Language Theory and Infinite Graphs

Colin Stirling
School of Informatics
University of Edinburgh

Characterising DPDA

- A DPDA can be converted into a determinized SDG, and configuration $p\alpha$ is converted into sum configuration $\mathrm{sum}(p\alpha)$ of the SDG where

  $\mathsf{G}^{\mathsf{c}}(p\alpha) \sim \mathsf{G}^{\mathsf{d}}(\mathrm{sum}(p\alpha))$

- (Conversely, a determinized SDG can be transformed into a DPDA)

- DPDA equivalence problem $=$ to deciding whether two admissible configurations of a determinized SDG are bisimulation equivalent

$$\alpha_1 + \ldots + \alpha_n \sim \beta_1 + \ldots + \beta_m \quad ?$$

Notation

- E, F, G, ... range over admissible configurations
- $+$ 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 $E \cup F$
- Also use sequential composition, if E and F are admissible, then EF is the admissible configuration $\{\beta\gamma : \beta \in E \text{ and } \gamma \in F\}$
- $E = E_1G_1 + \ldots + E_nG_n$ is a head/tail form, if the head $E_1 + \ldots + E_n$ is admissible and at least one $E_i \neq \emptyset$, and each tail $G_i \neq \emptyset$

  If we write $E = E_1G_1 + \ldots + E_nG_n$ then E is a head/tail form

Congruence

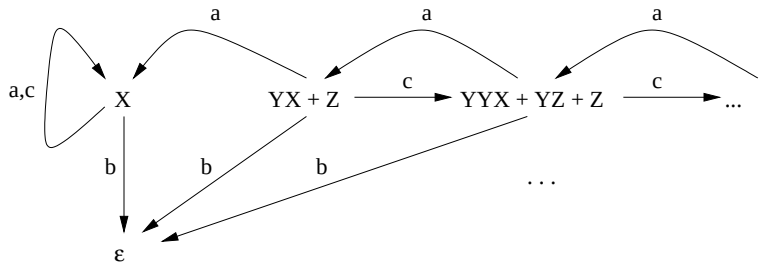
- Decision question $E \sim F$?
- for all j, $G_j \sim H_j$ iff $E_1G_1 + \ldots + E_nG_n \sim E_1H_1 + \ldots + E_nH_n$
- We can tear apart configurations and replace parts with equivalent parts

Notation

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

Example

$X \xrightarrow{a} X \quad Y \xrightarrow{a} \epsilon \quad Z \xrightarrow{a} \emptyset \quad X \xrightarrow{b} \epsilon \quad Y \xrightarrow{b} \emptyset$
$Z \xrightarrow{b} \epsilon \quad X \xrightarrow{c} X \quad Y \xrightarrow{c} YY \quad Z \xrightarrow{c} YZ + Z$



Interested in showing, for instance, that $YX + Z \sim YYX + YZ + Z$

DPDA decision procedure

- Given by a deterministic tableau proof system where goals are reduced to subgoals
- Goals and subgoals have the form $E \doteq F$, is $E \sim F$? E, F admissible
- Rules are (generalisations of) unfold, UNF, and balance rules,

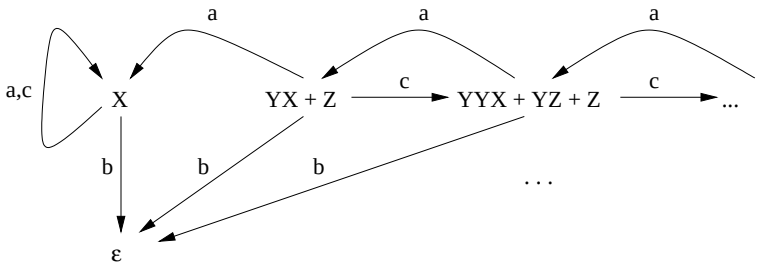
  BAL(L) and BAL(R). (CUT free)

Proof rules: UNF

$$\frac{E \dot{=} F}{(E \cdot a_1) \dot{=} (F \cdot a_1) \quad \ldots \quad (E \cdot a_k) \dot{=} (F \cdot a_k)} \; \mathsf{A} = \{a_1, \ldots, a_k\}$$

Example

$X \xrightarrow{a} X \quad Y \xrightarrow{a} \epsilon \quad Z \xrightarrow{a} \emptyset \quad X \xrightarrow{b} \epsilon \quad Y \xrightarrow{b} \emptyset$
$Z \xrightarrow{b} \epsilon \quad X \xrightarrow{c} X \quad Y \xrightarrow{c} YY \quad Z \xrightarrow{c} YZ + Z$



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

Proof rules: BAL(L)

$$\frac{\begin{array}{rcl} X_1H_1 + \ldots + X_kH_k & \dot{=} & F \\ \vdots & & \\ E_1H_1 + \ldots + E_kH_k & \dot{=} & F' \end{array}}{E_1(F \cdot w(X_1)) + \ldots + E_k(F \cdot w(X_k)) \;\dot{=}\; F'} \; \mathrm{C}$$

where C is the condition

1. Each $E_i \neq \varepsilon$ and at least one $H_i \neq \varepsilon$

2. There are $\max\{\, |w(X_i)| \;:\; E_i \neq \emptyset \text{ for } 1 \leq i \leq k\}$ applications of UNF between the top goal and the bottom goal, and no application of any other rule

Proof rules: BAL(R)

$$\frac{\begin{array}{lcl} F & \dot{=} & X_1 H_1 + \ldots + X_k H_k \\ & \vdots & \\ F' & \dot{=} & E_1 H_1 + \ldots + E_k H_k \end{array}}{F' \quad \dot{=} \quad E_1(F \cdot w(X_1)) + \ldots + E_k(F \cdot w(X_k))} \; \mathrm{C}$$

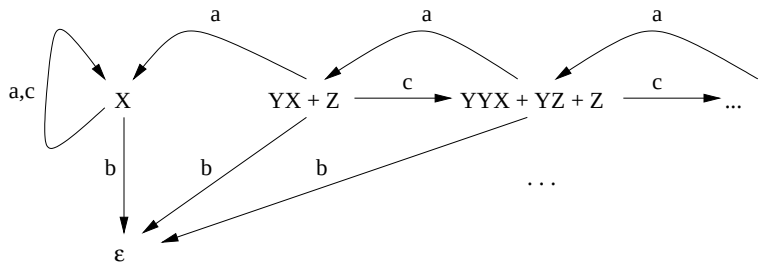
where C is the condition

1. Each $E_i \neq \varepsilon$ and at least one $H_i \neq \varepsilon$

2. There are $\max\{\, |w(X_i)| \;:\; E_i \neq \emptyset \text{ for } 1 \leq i \leq k\}$ applications of UNF between the top goal and the bottom goal, and no application of any other rule

Example

$X \xrightarrow{a} X \quad Y \xrightarrow{a} \epsilon \quad Z \xrightarrow{a} \emptyset \quad X \xrightarrow{b} \epsilon \quad Y \xrightarrow{b} \emptyset$
$Z \xrightarrow{b} \epsilon \quad X \xrightarrow{c} X \quad Y \xrightarrow{c} YY \quad Z \xrightarrow{c} YZ + Z$



$$\dfrac{\dfrac{}{YX + Z \doteq YYX + YZ + Z\ (F)}\ \text{UNF} \quad \dots \quad \dfrac{}{YYX + YZ + Z \doteq YYYX + YYZ + YZ + Z}\ \text{UNF}}{YYYX + YYZ + YZ + Z \doteq YYYX + YYZ + YZ + Z}\ \text{BAL(L)}$$

$w(Y) = a$, $w(Z) = b$ and $(F \cdot w(Y)) = YX + Z$, $(F \cdot w(Z)) = \epsilon$

Procedure

- Start with an initial goal, $E \doteq F$, and deterministically build a proof tree by applying the tableau rules (there is an ordering on the rules)
- Goals are thereby reduced to subgoals
- Need to define final goals. Rules are not applied to final goals

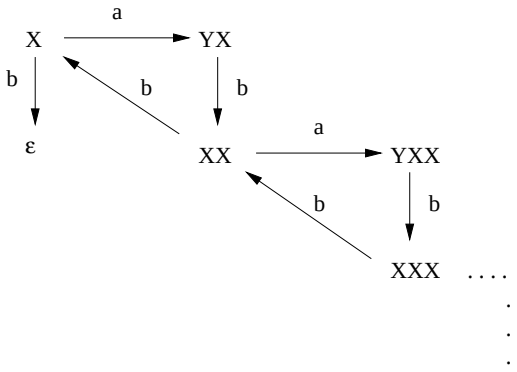
A successful tableau is a finite proof tree all of whose leaves are successful final goals

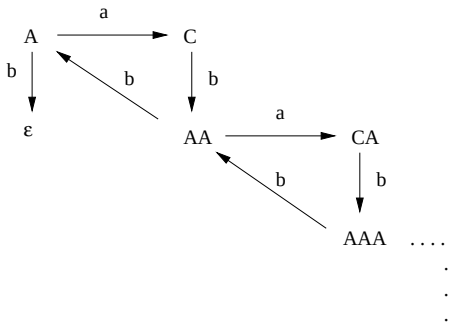
Otherwise a tableau is unsuccessful

Unsuccessful final goals

$E \doteq F$ (exactly one of E, F is $\emptyset$)

Infinite proof tree ?





$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{XX^5 \doteq AA^5}}{YXX^5 \doteq CA^5}\,\text{BAL(L)}}{YXA^5 \doteq CA^5}\,\text{UNF}}{\emptyset \doteq \emptyset}\qquad \dfrac{\dfrac{\dfrac{}{XXA^5 \doteq AA^6}}{YXXA^5 \doteq CA^6}\,\text{BAL(L)}}{YXA^6 \doteq CA^6}\,\text{UNF}}{X^5 \doteq A^5}\,\text{UNF}}{XA^5 \doteq A^6}\,\text{UNF}$$

Extensions

- If $E = E_1G_1 + \ldots + E_nG_n$, $F = F_1H_1 + \ldots + F_mH_m$, then

  F is a tail extension of E provided that for each $i : 1 \leq i \leq m$,

  $$H_i = K_1^iG_1 + \ldots + K_n^iG_n$$

- The extension e is the m-tuple $(K_1^1 + \ldots + K_n^1, \ldots, K_1^m + \ldots + K_n^m)$ without the G_is, the width of e is m, and F is said to extend E with e

- Extensions are matrices, written in a linear notation

- If $E = E_1G_1 + \ldots + E_nG_n$ and $F = F_1G_1 + \ldots + F_nG_n$, then E extends F by $e = (\varepsilon + \emptyset + \ldots + \emptyset, \ldots, \emptyset + \emptyset + \ldots + \varepsilon)$. Identity extension (ε)

Composing extensions

If

$E = E_1G_1 + \ldots + E_lG_l$ and

$E' = E'_1G'_1 + \ldots + E'_mG'_m$ and

$E'' = E''_1G''_1 + \ldots + E''_nG''_n$ and

E' extends E by $e = (J^1_1 + \ldots + J^1_l, \ldots, J^m_1 + \ldots + J^m_l)$ and

E'' extends E' by $f = (K^1_1 + \ldots + K^1_m, \ldots, K^n_1 + \ldots + K^n_m)$,

then E'' extends E by $ef = (H^1_1 + \ldots + H^1_l, \ldots, H^n_1 + \ldots + H^n_l)$

where $H^i_j = K^i_1J^1_j + \ldots + K^i_mJ^m_j$

Example

$$
\begin{array}{lll}
E = YG_1 + ZG_2 & \text{where} & G_1 = X \text{ and } G_2 = \varepsilon \\
E' = YG_1' + ZG_2' & \text{where} & G_1' = YX + Z \text{ and } G_2' = \varepsilon \\
E'' = YG_1'' + ZG_2'' & \text{where} & G_1'' = YYX + YZ + Z \text{ and } G_2'' = \varepsilon
\end{array}
$$

E' extends E by $e = (Y + Z, \emptyset + \varepsilon)$ and

E'' extends E' by $f = e = (Y + Z, \emptyset + \varepsilon)$

Therefore, E'' extends E by $ef = (YY + (YZ + Z), \emptyset + \varepsilon)$.

Notation

Assume goals with the same heads

$$\text{goal g} \quad (E)\ E_1G_1 + \ldots + E_nG_n \doteq F_1G_1 + \ldots + F_nG_n\ (F)$$

$$\text{goal h} \quad (E')\ E_1H_1 + \ldots + E_nH_n \doteq F_1H_1 + \ldots + F_nH_n\ (F')$$

h extends g by e, if E' extends E by e (and F' extends F by e)

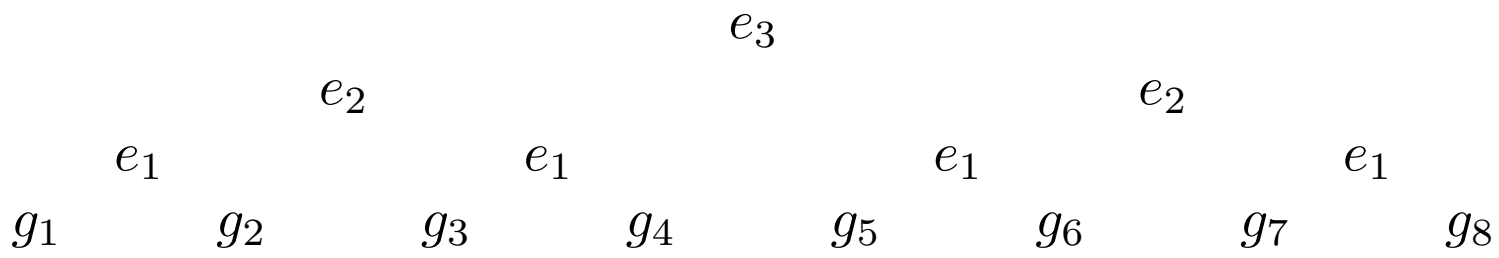
Repeating patterns

Width is 3. Assume $g(i)$, $h(i)$, $i : 1 \leq i \leq 8$

$$E_1G_1^i + E_2G_2^i + E_3G_3^i \doteq F_1G_1^i + F_2G_2^i + F_3G_3^i$$

$$E_1H_1^i + E_2H_2^i + E_3H_3^i \doteq F_1H_1^i + F_2H_2^i + F_3H_3^i$$

Assume extensions e_1, e_2 and e_3 for $g(i)$ and $h(i)$



If each $g(i)$, $1 \leq i \leq 8$, and each $h(i)$, $1 \leq i < 8$, m-true then $h(8)$ is m-true

General case

If there are two families of goals $g(i)$, $h(i)$, $1 \leq i \leq 2^n$

$$E_1 G_1^i + \ldots + E_n G_n^i \doteq F_1 G_1^i + \ldots + F_n G_n^i$$

$$E_1 H_1^i + \ldots + E_n H_n^i \doteq F_1 H_1^i + \ldots + F_n H_n^i$$

and extensions $e_1, \ldots, e_n$ such that for each e_j and $i \geq 0$

$g(2^j i + 2^{j-1} + 1)$ extends $g(2^j i + 2^{j-1})$ by e_j
$h(2^j i + 2^{j-1} + 1)$ extends $h(2^j i + 2^{j-1})$ by e_j.

and each $g(i)$, $i : 1 \leq i \leq 2^n$, and each $h(j)$, $j : 1 \leq j < 2^n$, is m-true, then $h(2^n)$ is m-true

Successful final goals

$E \doteq F$ the root goal

$\vdots$

$\vdots$ there are goals $g(i)$ $h(i) \ldots$

$\vdots$ $h(2^n)$ is $E' \doteq F'$

$E \doteq E$ $E' \doteq F'$

Decidability

Propositions

- Every tableau is finite
- $E \sim F$ iff the tableau with root $E \doteq F$ is successful

Complexity upper bound is primitive recursive (because of extensions). The number of goals involved may grow with the number of states when applying extension theorem as it is width of a goal that counts

Corollary: $E \sim F$ iff $E \sim_{f(n)} F$

where f is the primitive recursive function, n is the size of the DPDA, E and F

Higher-order pushdown automata

Basic transitions of the form for order n

$pX \xrightarrow{a} q\,\mathsf{Op}$ where $\mathsf{Op} \in \{\mathrm{swap}(\alpha), \mathrm{push}(i), \mathrm{pop}(i)\ 1 < i \leq n\}$

Stacks of stacks of stacks ...

Example at order 2

$$p'Z \xrightarrow{a} pZ$$

$$pZ \xrightarrow{a} pXZ \qquad pX \xrightarrow{a} pXX \qquad pX \xrightarrow{b} q\,\mathrm{push}$$

$$qX \xrightarrow{b} q\epsilon \qquad qZ \xrightarrow{c} r\,\mathrm{pop}$$

$$rX \xrightarrow{c} r\epsilon \qquad rZ \xrightarrow{\epsilon} r\epsilon$$

Results/Open Questions

- The languages generable by order 2 pushdown automata are indexed languages (mildly context-sensitive languages) introduced by Aho 1968

- The monadic second-order theory of any graph generated by a higher-order automaton is decidable (Knapik, Niwinski and Urzyczyn 2002)

  Proof nice mixture of Rabin's theorem and geometry of interaction

- Are languages generable by order n, $n > 2$, context-sensitive?

- Is language equivalence decidable for determinisitic higher-order pda ?

  Intimately related to higher-order schema problem open since 1960s

Thanks for listening