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

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

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 $X \in \mathsf{S}$, $a \in \mathsf{A}$ and $\alpha \in \mathsf{S}^*$

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: $X \xrightarrow{a} \epsilon \in$ T
- Identify stack symbols with norm 2: $X \xrightarrow{a} Z \in$ T and Z has norm 1
- Identify stack symbols with norm 3: $X \xrightarrow{a} Z \in$ T and Z has norm 2 or $X \xrightarrow{a} YZ \in$ T and both Y and Z have norm 1
- $\ldots$
- 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 $X \xrightarrow{a} \alpha \in$ T and $X \xrightarrow{a} \beta \in$ T then $\alpha = \beta$

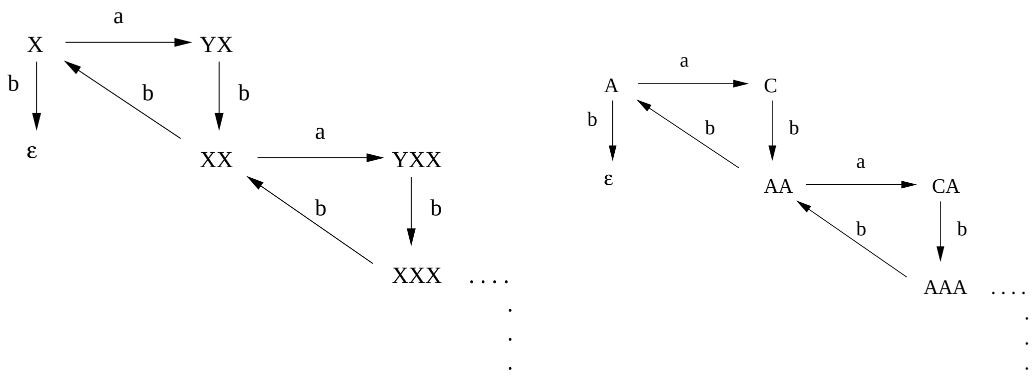
Recognition power of simple grammars is less than that of DPDA:

$L = \{a^n b^n \,:\, n > 0\} \cup \{a^n c \,:\, n > 0\}$ 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 $w(C) = bbb$

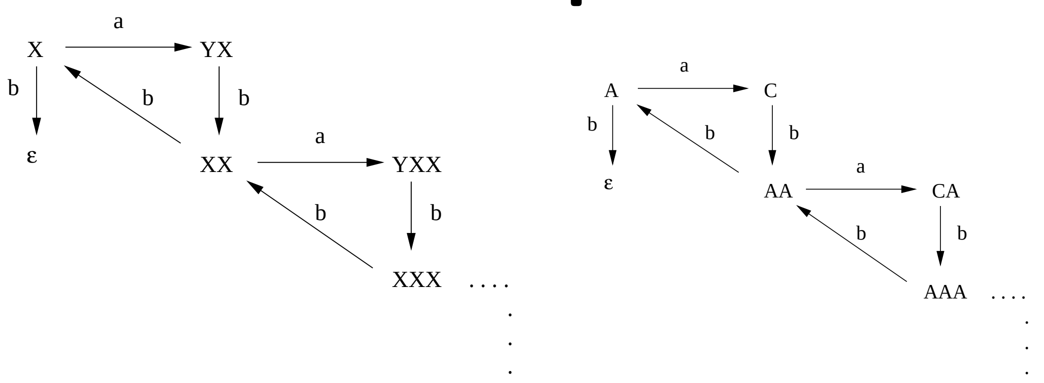
Maximum norm $\mathrm{M} = 3$

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

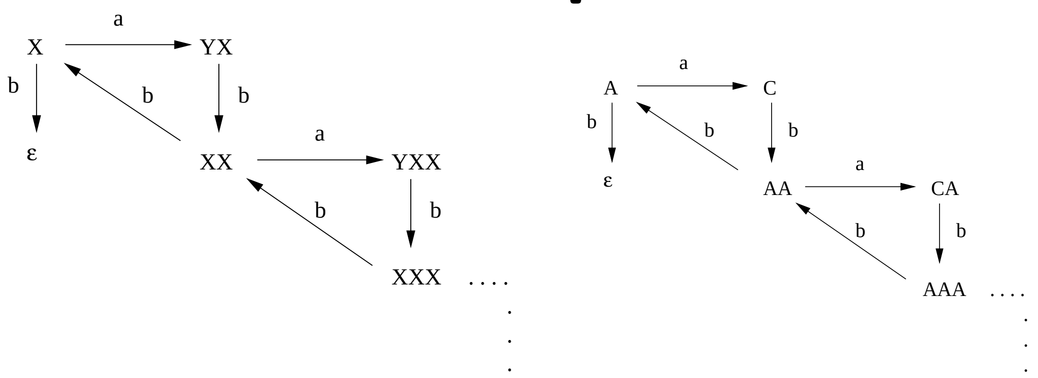
Given $\alpha, \beta \in \mathsf{S}^*$, is $L(\alpha) = L(\beta)$?

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)

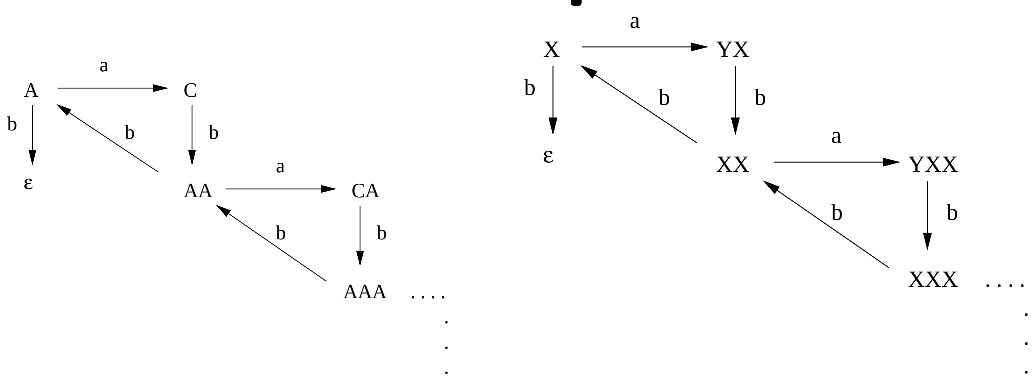
Goal, $\alpha \doteq \beta$ reduces to subgoals $(\alpha \cdot a) \doteq (\beta \cdot a)$ for each $a \in \mathsf{A}$

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

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



$$\dfrac{A \doteq X}{\dfrac{C \doteq YX}{\emptyset \doteq \emptyset \qquad \dfrac{AA \doteq XX}{CA \doteq YXX \qquad A \doteq X}\ \mathrm{UNF}}\ \mathrm{UNF} \qquad \epsilon \doteq \epsilon}\ \mathrm{UNF}$$

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)

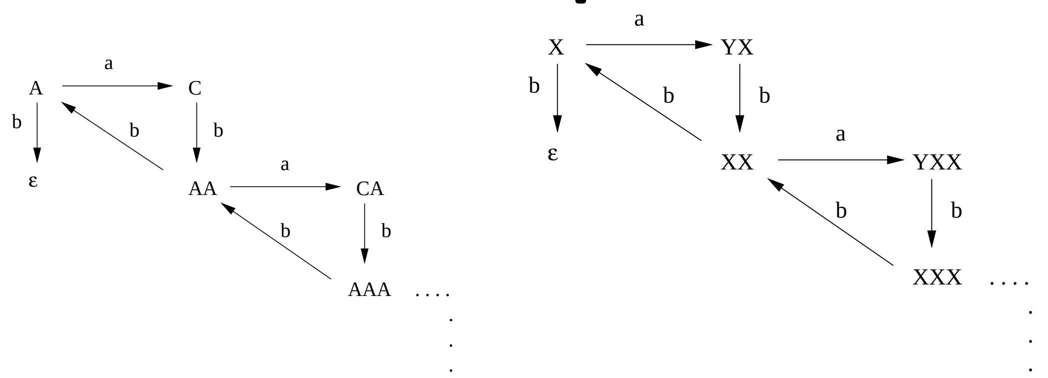
$$\frac{\begin{array}{rcl} X\alpha & \doteq & \beta \\ & \vdots & \\ \alpha'\alpha & \doteq & \beta' \end{array}}{\alpha'(\beta \cdot w(X)) \doteq \beta'}\ \mathsf{C} \qquad \frac{\begin{array}{rcl} \beta & \doteq & X\alpha \\ & \vdots & \\ \beta' & \doteq & \alpha'\alpha \end{array}}{\beta' \doteq \alpha'(\beta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

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, $X\alpha \doteq \beta$ ($\beta \doteq X\alpha$), and the bottom goal, $\alpha'\alpha \doteq \beta'$ ($\beta' \doteq \alpha'\alpha$), and no applications of any other rule

Example



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

$w(A) = b$ 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

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

Unsuccessful final goals

$\alpha \doteq \beta$ (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

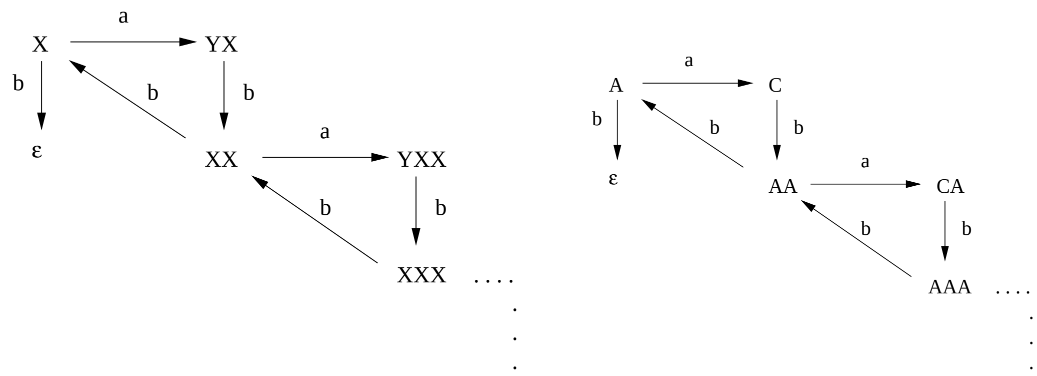
We assume that a BAL rule applies to a goal using the premise above which is the closest

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



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

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

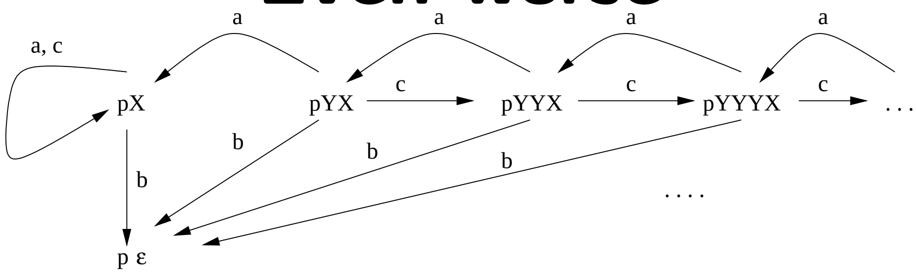
- How can we tear apart DPDA (stable) configurations $p\alpha$ and replace parts with parts? What is a part of a configuration?

  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