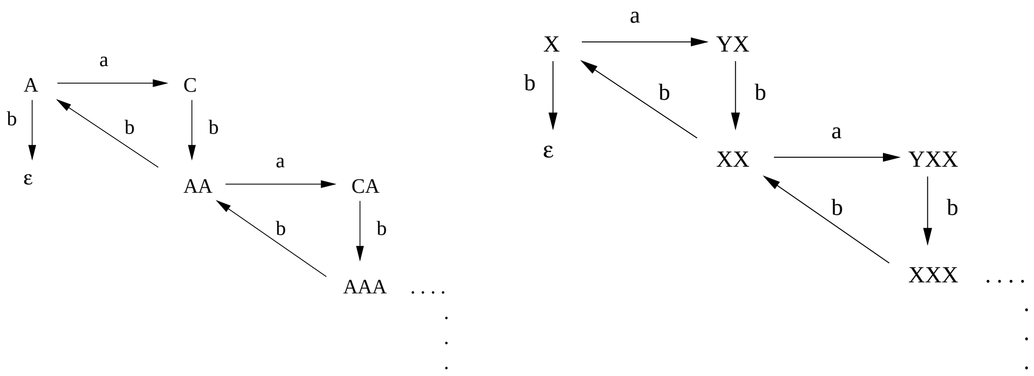
Language Theory and Infinite Graphs

Colin Stirling
School of Informatics
University of Edinburgh

Example



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

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

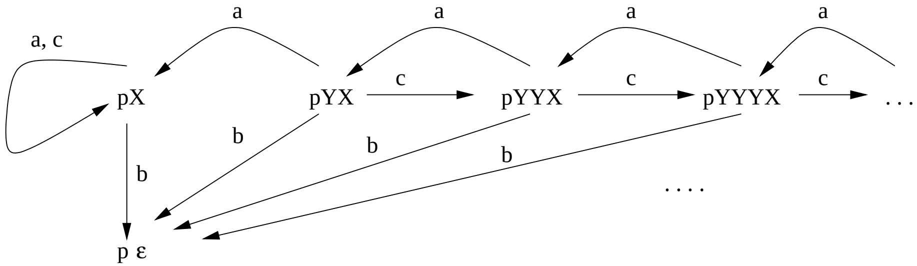
- How can we tear apart DPDA (stable) configurations $p\alpha$ and replace parts with parts? What is a part of a configuration?

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

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

Example

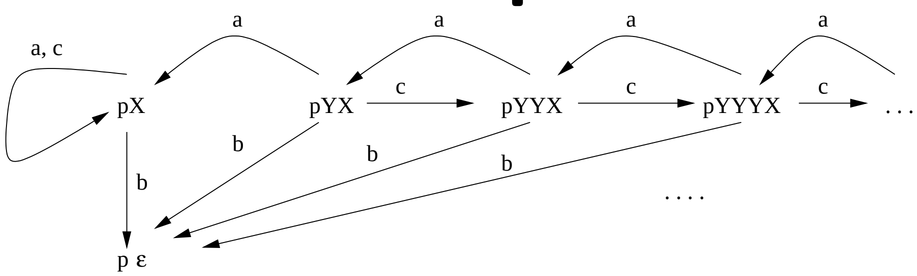
$$
\begin{array}{ccccccc}
q\epsilon & \xleftarrow{b} & qX & \xleftarrow{b} & qXX & \xleftarrow{b} & \ldots \\
\uparrow b & & \uparrow b & & \uparrow b & & \\
pX & \xrightarrow{a} & pXX & \xrightarrow{a} & pXXX & \xrightarrow{a} & \ldots \\
\downarrow b & & \downarrow b & & \downarrow b & & \\
r\epsilon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

- $[pXq] \xrightarrow{a} [pXp][pXq] \ldots$
- $[pXq] \xrightarrow{a} [pXq][qXq] \ldots$
- 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

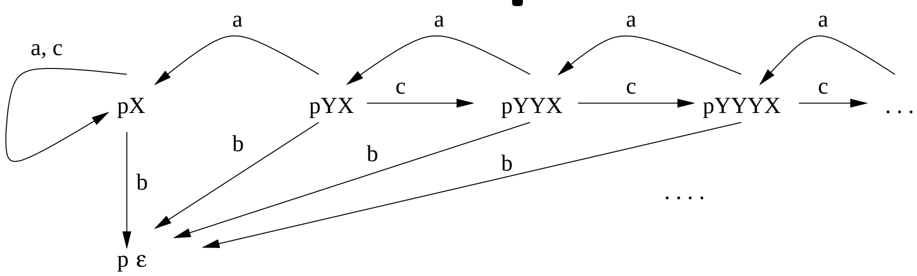
Example



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

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 $[pXq] \xrightarrow{a} [rYq]$

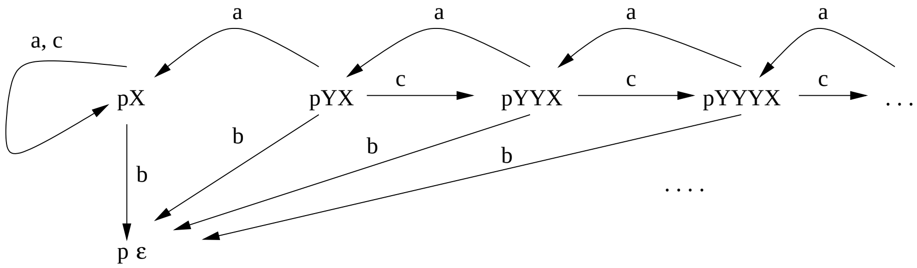
  $pX \xrightarrow{a} rYZ$ is $[pXq] \xrightarrow{a} \sum\{[rYp'][p'Zq] \,:\, p' \in \mathsf{P}\}$

- 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



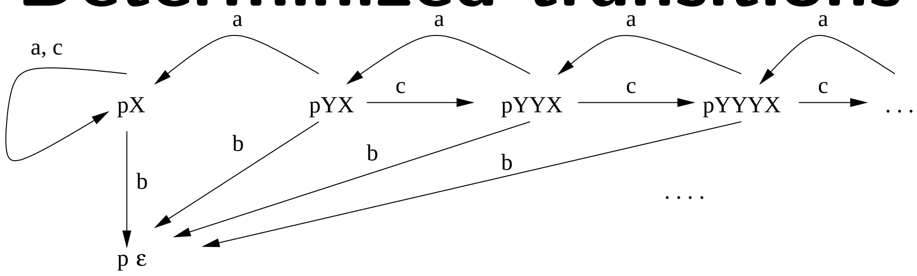
$$[pXp] \xrightarrow{a} [pXp] \qquad [pXp] \xrightarrow{b} \epsilon \qquad [pXp] \xrightarrow{c} [pXp]$$

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

$$[pYr] \xrightarrow{c} [pYp][pYr] \qquad [pYr] \xrightarrow{c} [pYr]$$

- 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 $\delta \in \mathsf{S}^*$, $\alpha = \delta X \alpha'$, $\beta = \delta Y \beta'$, $X \equiv Y$ 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

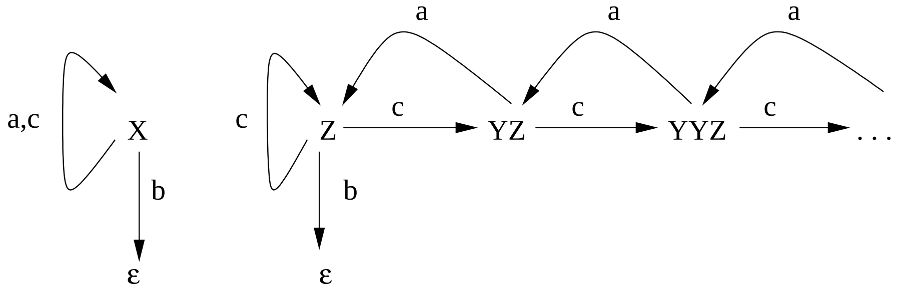
$$1.\quad \text{if} \quad \begin{array}{c} X \xrightarrow{a} \alpha \in \mathsf{T} \\ ||| \\ Y \xrightarrow{a} \beta \in \mathsf{T} \end{array} \quad \text{then} \quad \begin{array}{c} \alpha \\ ||| \\ \beta \end{array}$$

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

A PDG with $\equiv$ is strict determinisitic, 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

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

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

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

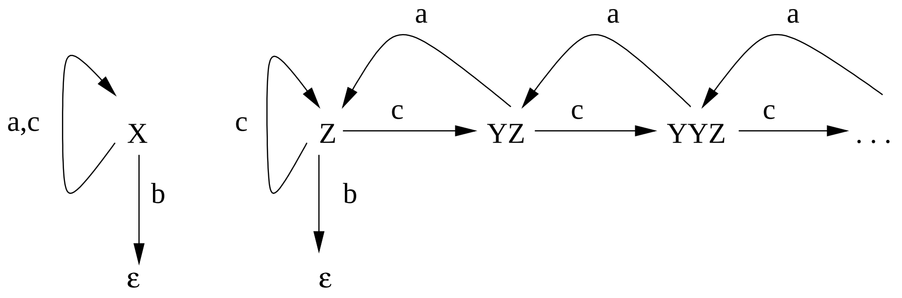
  If $\alpha_1 + \ldots + \alpha_n$ admissible then for all w the sum configuration

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

  is admissible

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

$$
\begin{array}{ll}
\text{If } X_i \xrightarrow{a} \alpha_1^i + \ldots + \alpha_{n_i}^i & \text{for each } i : 1 \leq i \leq m \text{ then} \\
X_1\beta_1 + \ldots + X_m\beta_m \xrightarrow{a} & \alpha_1^1\beta_1 + \ldots + \alpha_{n_1}^1\beta_1 \quad + \\
 & \ldots \qquad\qquad \ldots \qquad\quad + \\
 & \alpha_1^m\beta_m + \ldots + \alpha_{n_m}^m\beta_m
\end{array}
$$

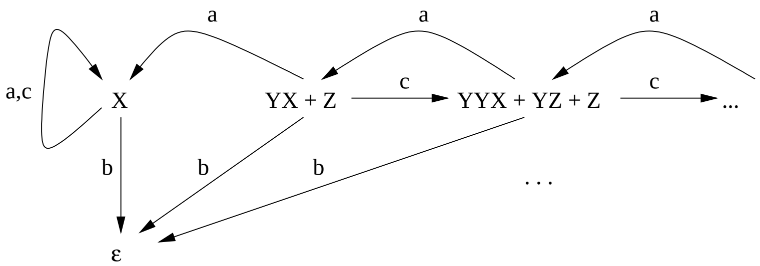
- Determinized graph $\mathsf{G}^d(\alpha_1 + \ldots + \alpha_n)$ using T^d and the extended prefix rule

Example

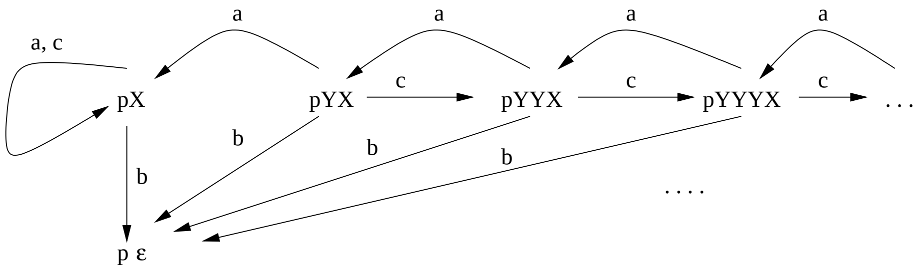
S $= \{X, Y, Z\}$, A $= \{a, b, c\}$ and the partition of S $= \{\{X\}, \{Y, Z\}\}$. And T^d is

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

G^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 $\mathsf{sum}(p\alpha)$ 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 $E = E_1G_1 + \ldots + E_nG_n$
  - 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 $E \sim_{m+1} E_1H_1 + \ldots + E_nH_n$
  - If each $H_i \neq \emptyset$ and for each j such that $E_j \neq \emptyset$, $H_j \sim G_j$, then $E \sim E_1H_1 + \ldots + E_nH_n$

- 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 $E \xrightarrow{u} F$

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)