More PDAs

CS154
Chris Pollett
Mar 21, 2007.

Outline

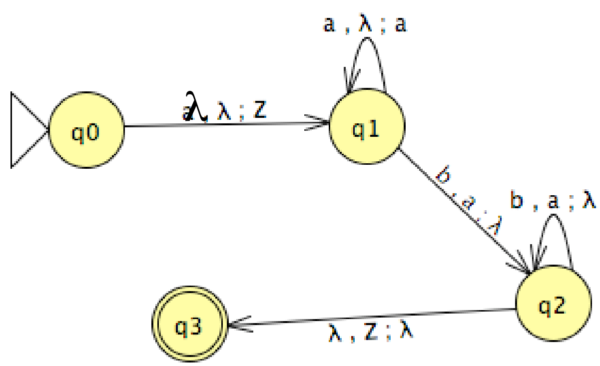
- PDA languages are CFL
- Deterministic PDAs
- Grammars for DCFLs

PDA recognizes => CFL

- Let P be a PDA. We want to make a CFG G that generates the same language.
- For each pair of states p,q in P we will have in G a variable A_{pq}. This variable will be able to generate all strings that can take P in state p with the empty stack to state q with the empty stack.
- To simplify the problem we will assume P has been modified so that:
  - it has a single accept state
  - it empties its stack (except start of stack symbol) before accepting
  - each transition either pushes a symbol onto the stack or pops one off of the stack (but not both and not neither). (We might add states to make our machine have this property).
- G will have rules $A_{pp} \dashrightarrow \lambda$ for each state p of P; $A_{pq} \dashrightarrow aA_{rs}b$ for each p,q, r,s such that $\delta(p,a,\lambda)$ contains (r,t) and $\delta(s,b,t)$ contains (q, λ), and $A_{pq} \rightarrow A_{pr}A_{rq}$ for any state r.
- The start variable of G will be $A_{q0,qaccept}$.

Example

- Consider the machine:



- It recognizes the language $\{a^n b^n \mid n>0\}$.
- It has a single accept state and each transition either pushes or pops a symbol, so we can apply the construction.
- This machine empties the stack except for the start of stack symbol.
- The start variable given by the construction will be A_{q0q3}. We'll abbreviate this as A_{03}.
- Many of the rules the construction would give are completely useless; nevertheless, one can check it does produce the rules A_{03}--> $\lambda A_{12}\,\lambda$, A_{12}-->$aA_{12}b$, A_{12}-->$aA_{11}b$ and A_{11}--> λ.

DPDAs

Defn. A PDA is called a **deterministic PDA (DPDA)** if:

(1) $\delta(q,a,b)$ only contains one element.

(2) if δ (q, λ, b) is not empty, then $\delta(q,c,b)$ must be empty for every c in Σ.

- These conditions ensure there is at most one move, in any fixed state with the same top of stack.
- Notice unlike DFAs we still allow λ transitions.
- A language is **DCFL** (a **deterministic context free language**) if it is recognized by a DPDA.

Example

- $L = \{a^n b^n \mid n \geq 0\}$ is a DCFL. We can take M to be: $(\{q0, q1, q2\}, \{a,b\}, \{Z,1\}, \delta, q0, Z, \{q0\})$ where:

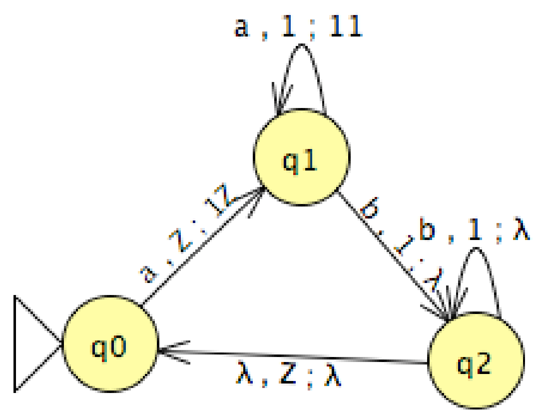
  $\delta(q0,a, Z) = \{(q1, 1Z)\}$,

  $\delta(q1, a, 1) = \{(q1, 11)\}$,

  $\delta(q1, b, 1) = \{(q2, \lambda)\}$,

  $\delta(q2, b, 1) = \{(q2, \lambda)\}$,

  $\delta(q2, \lambda, 0) = \{(q0, \lambda)\}$



- I am using Z as the initial stack symbol, as this is what JFLAP uses.

Example

Consider L the union of the language $L1=\{a^n b^n \mid n \geq 0\}$ and $L2= \{a^n b^{2n} \mid n \geq 0\}$.

- Each of these languages individually is DCFL.
- There union is context free. To see this take a CFG for each with start symbols respectively S1 and S2. Then add the new start symbol S and rules S-->S1 |S2.
- It turns out L is not DCFL.

JFLAP Examples

- We've already mentioned JFLAP has tools for grammars.
- On its main button panel it also has a button "Pushdown Automata"
- This allows the user to create pushdown automata in much the same way as DFAs are made in JFLAP.
- When you choose the "Test" menu to run an automata on some inputs you will notice that the starting stack symbol is always Z.
- Next day we'll look at how JFLAP converts PDAs to CFGs