Chapter 7: Pushdown Automata (PDA)*

Peter Cappello
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106
cappello@cs.ucsb.edu

- Please read the corresponding chapter before attending this lecture.
- These notes are supplemented with figures, and material that arises during the lecture in response to questions.
- Please report any errors in these notes to `cappello@cs.ucsb.edu`. I'll fix them immediately.

*Based on **An Introduction to Formal Languages and Automata**, 3rd Ed., Peter Linz, Jones and Bartlett Publishers, Inc.

7.1 Nondeterministic Pushdown Automata

Definition of a Pushdown Automaton (Illustrate schematic of PDA.)

Def. 7.1: A **nondeterministic pushdown acceptor (NPDA)** is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$, where

Q is a finite set of states of the control unit,

Σ is a finite input alphabet,

Γ is a finite **stack alphabet**,

δ**:** $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \mapsto$ *finite* subsets of $Q \times \Gamma^*$ is the transition function,

q_0 is the initial state,

$z \in \Gamma$ is the **stack start symbol**,

$F \subseteq Q$ is the set of final states.

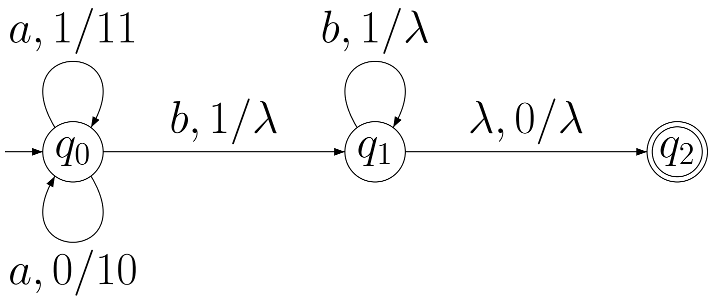
- The range of δ is a set of ordered pairs.
- The 1st component of these ordered pairs is the successor state.
- The second component is a sequence of stack symbols that replace the top stack symbol.

  For example, if the current stack symbol is α and the ordered pair specified $(q, \alpha\beta\gamma)$, then, in effect, we execute the following sequence:

  pop
  push(γ)
  push(β)
  push(α).
- The set $Q \times \Gamma^*$ is infinite, and hence has infinite subsets.

  We disallow the set of successor pairs to be infinite.

Example: Let $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_3\})$ be a NPDA with δ indicated diagrammatically below.



- Does M accept λ?
- Does M accept a?
- Does M accept ab?
- What language does M accept?

- An **instantaneous description (ID)** gives all the relevant information about the current state of an NPDA:
  - The current state;
  - The unread portion of the input;
  - The current contents of the stack.
- The operation of an NPDA can be depicted as a sequence of IDs, starting from the initial ID: (q_0, w, z_0), where the input is w and z_0 is the start stack symbol.
- A move from ID (q_i, aw, bx) to ID (q_j, w, yx) is denoted

$$(q_i, aw, bx) \vdash (q_j, w, yx),$$

  and is possible if and only if $(q_j, y) \in \delta(q_i, a, b)$.
- The notation $ID_i \vdash^* ID_j$ denotes a sequence of moves from ID_i to ID_j.

Def. 7.2: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ be a NPDA.
The language accepted by M is

$$L(M) = \{w \in \Sigma^* : (q_0, w, z) \vdash_M^* (p, \lambda, u), p \in F, u \in \Gamma^*\}.$$

The final stack content, u, is irrelevant.

Example 7.4:

- $L = \{ww^R : w \in \{a, b\}^+\}$.
- The algorithm:
  1. Read the symbols of w; push them onto the stack;
  2. Guess that the last symbol of w has been read/pushed.
  3. Read the remainder of the input, checking that each symbol matches the top of the stack, which is then popped.
- Illustrate the NPDA diagrammatically.