Pushdown Automata

A *pushdown automaton*, or PDA, extends the ε-NFA model by adding a stack with its own alphabet Γ (which may be different from Σ). Naturally, only the topmost symbol on the stack is visible.

Transition notation for the stack:

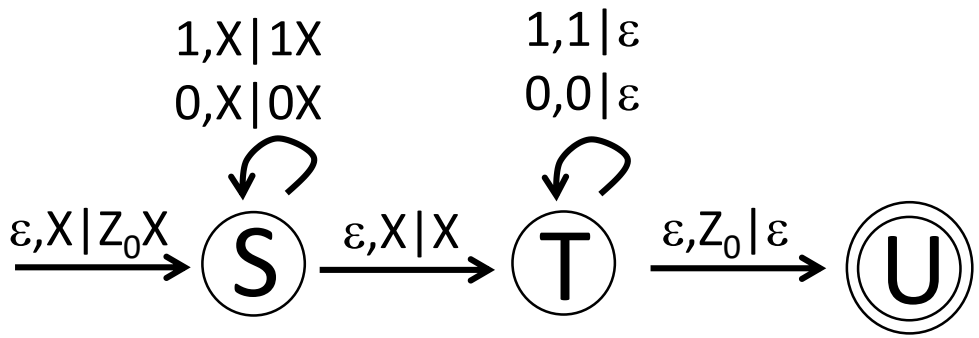
$\xrightarrow{a,b|cb}$ means: on input a with b on top of the stack, push c (on top of b).

$\xrightarrow{a,b|c}$ means: on input a with b on top of the stack, pop the stack and push c.

$\xrightarrow{a,b|\varepsilon}$ means: on input a with b on top of the stack, pop the stack.

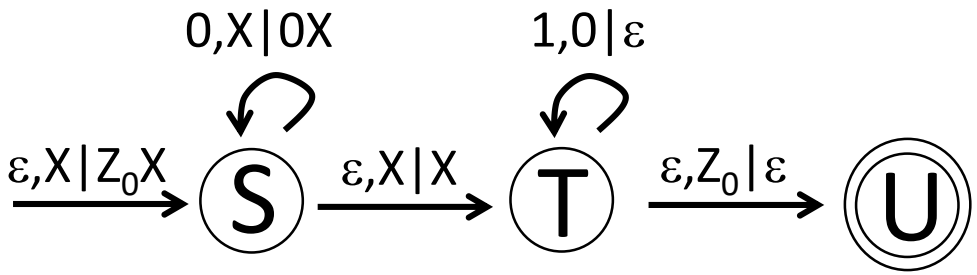
We will use the symbol Z_0 for the stack bottom (it marks the empty stack) and X as a placeholder for anything on the stack.

Example: the following PDA accepts strings in (0+1)* that are even-length palindromes



This automaton accepts strings that get to state U after consuming all of their input. Note that if it starts with an empty stack the stack will be empty at the end of the input.

Here is another example. This automaton accepts $\{0^n1^n | n>=0\}$

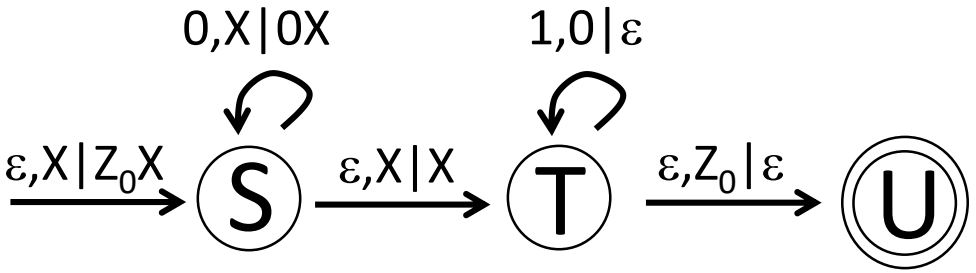


We need a more formal and deterministic way to think about PDA computations.

An *instantaneous description* (ID) is a triple (q, w, γ) where

- q is a state
- w is a string (the portion of the input not yet used)
- γ is a string of stack symbols (the complete contents of the stack, with the top on the left)

Here is an ID analysis of this PDA for the input 0011:



$(S,0011,Z_0) \longrightarrow (T,0011,Z_0) \longrightarrow (U,0011,Z_0)$

$(S,011,0Z_0) \longrightarrow (T,011,0Z_0)$

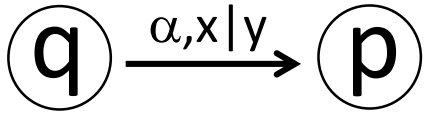
$(S,11,00Z_0) \longrightarrow (T,11,00Z_0)$

$(T,1,0Z_0)$

$(T,\varepsilon,Z_0) \longrightarrow (U,\varepsilon,\varepsilon)$ accept

One step in such an ID analysis is $(q,\alpha w,x\beta) \longrightarrow (p,w,y\beta)$

This is valid if the PDA has a transition



We write $(q,w_1,\alpha) \xrightarrow{*} (p,w_2,\beta)$ if there is a sequence of steps that take the PDA from the first ID to the second.

Formally a PDA is a 7-tuple $(\Sigma,Q,\delta,s,F,\Gamma,Z_0)$ where

- Σ,Q,s,F have the same meanings as with DFAs
- δ is our configuration transformation function
- Γ is the alphabet of stack symbols
- Z_0 is the stack bottom

There are two commonly used definitions of what it means for the PDA to accept a string:

Acceptance by final state: If P is the PDA $(\Sigma,Q,\delta,s,F,\Gamma,Z_0)$ then $\mathcal{F}(P)$ = $\{w|$there is $q\in F$ and $\alpha\in\Gamma^*$ so that $(s,w,Z_0)\xrightarrow{*}(q,\varepsilon,\alpha)\}$

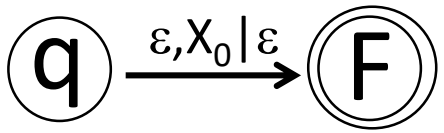
Acceptance by empty stack: If P is the PDA $(\Sigma,Q,\delta,s,F,\Gamma,Z_0)$ then $\mathcal{E}(P)$ = $\{w|$there is some state q so that $(s,w,Z_0)\xrightarrow{*}(q,\varepsilon,\varepsilon)\}$

For a given automaton P, $\mathcal{E}$(P) and $\mathcal{F}$(P) are not necessarily the same. However, the languages that can be accepted by empty stack are the same as those that can be acceped by final state:

Theorem 1: Start with PDA P. Then there is a PDA P' where $\mathcal{F}$(P')=$\mathcal{E}$(P).

Theorem 2: Start with PDA P. Then there is a PDA P' where $\mathcal{E}$(P')=$\mathcal{F}$(P).

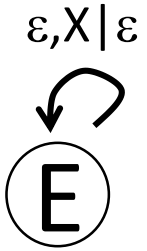
Theorem 1: Start with PDA P. Then there is a PDA P' where $\mathcal{F}(P')=\mathcal{E}(P)$.
Proof: To make P', start with P. Create a new start state s' which pushes a new stack symbol X_0 onto the stack before Z_0. Make a new final state F. For each state q of P add a transition



If P ever accepts a string by emptying its stack, P' can transition to its final state F.
On the other hand, if P' ever accepts a string by arriving in state F, then at the end of the input there must be only X_0 on the stack, so P must have emptied its stack.

Theorem 2: Start with PDA P. Then there is a PDA P' where $\mathcal{E}(P')=\mathcal{F}(P)$.

Proof: This is easy. Start with P' the same as P. Give P' a new state E that empties the stack:



and add an ε-transition from every final state to E. String w can take P to a final state if and only if w empties the stack of P'.