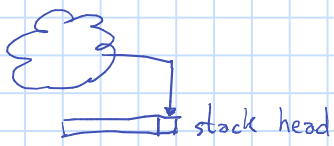
L7.

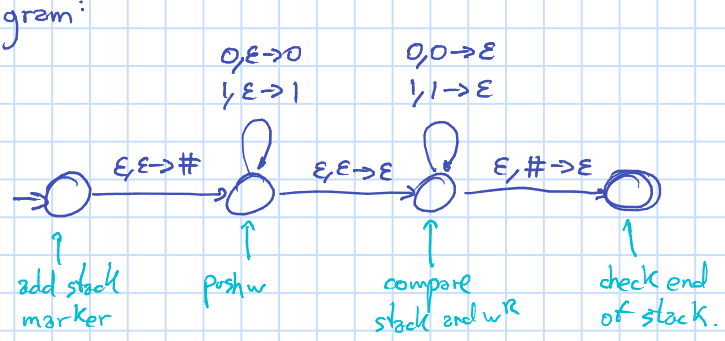
Pushdown Automata



e.g. $L = \{ww^R\}$. (even-length palindromes)

Informally: push w, then compare stack with w^R.
obs we don't know when w ends, we must guess that nondeterministically.

Diagram:



* if we add $(0, \varepsilon \to \varepsilon)$ and $(1, \varepsilon \to \varepsilon)$ between push and compare, then we recognize odd-length palindromes too.

Context-Free Grammars.

* Analogue of Regular expressions for PDAs.
* Useful in compiler design, for parsing.

A grammar is a set of substitution rules, where we replace a variable with other variables and characters.

Formally: (V, Σ, R, S).

V variables. Finite set.
Σ alphabet. Finite set
| disjoint.

R rules (types $(V, (V \cup \Sigma)^*)$).

$S \in V$ initial variable.

A variable may appear in the LHS of multiple rules (usually we collect rules with the same LHS variable, separated by |).

We choose which variable to expand and which rule nondeterministically.

Formally, a computation is a sequence of strings

$s_0, s_1, \dots s_n$ st $s_0 = S$, $s_i \in (V \cup \Sigma)^*$, s_i is the result of expanding one variable in s_{i-1}, and $s_n \in \Sigma^*$. We say s_n is accepted by the grammar.

e.g. $L = \{$correct parentheses$\}$

$S \to (S) \mid SS \mid \varepsilon$

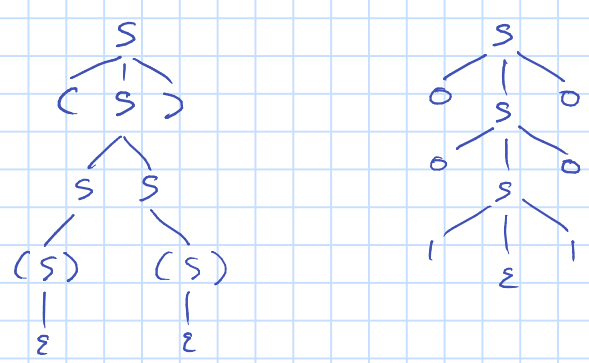
$S \to (S) \to (SS) \to ((S)S) \to (()S) \to$
$\to (()(S)) \to (()())$.

e.g. $L = \{$even-length palindromes$\}$.

$S \to 0S0 \mid 1S1 \mid \varepsilon$

$S \to 0S0 \to 00S00 \to 001S100 \to 001100$

We can represent computations as parse trees.



Not a coincidence that PDAs and grammars recognize these languages: they are equivalent.

Th: A language is recognized by a CFG iff it is recognized by a PDA.

proof: See Sipser.