CFG3: Push-down Automata

Context-Free Grammars

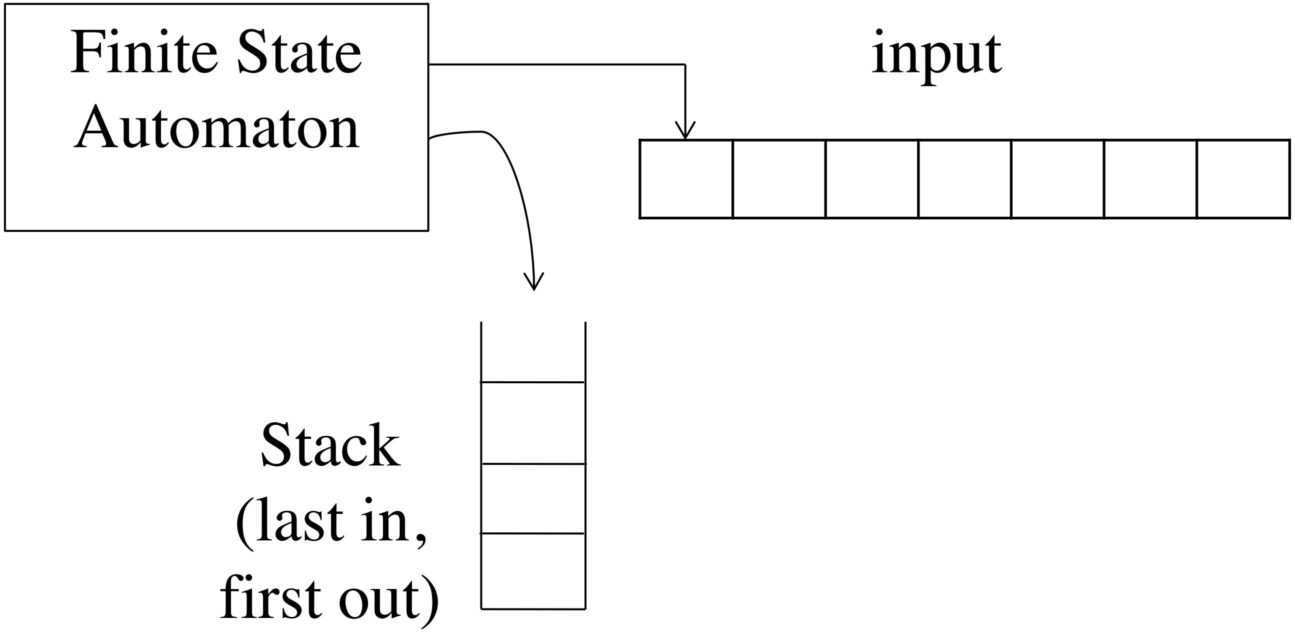
CMPT 379: Compilers

Instructor: Anoop Sarkar

anoopsarkar.github.io/compilers-class

Context-free languages and Pushdown Automata

- Recall that for each regular language there was an equivalent finite-state automaton
- The FSA was used as a recognizer of the regular language
- For each context-free language there is also an automaton that recognizes it: called a **pushdown automaton (pda)**



Finite State Automaton

input

Stack (last in, first out)

Context-free languages and Pushdown Automata

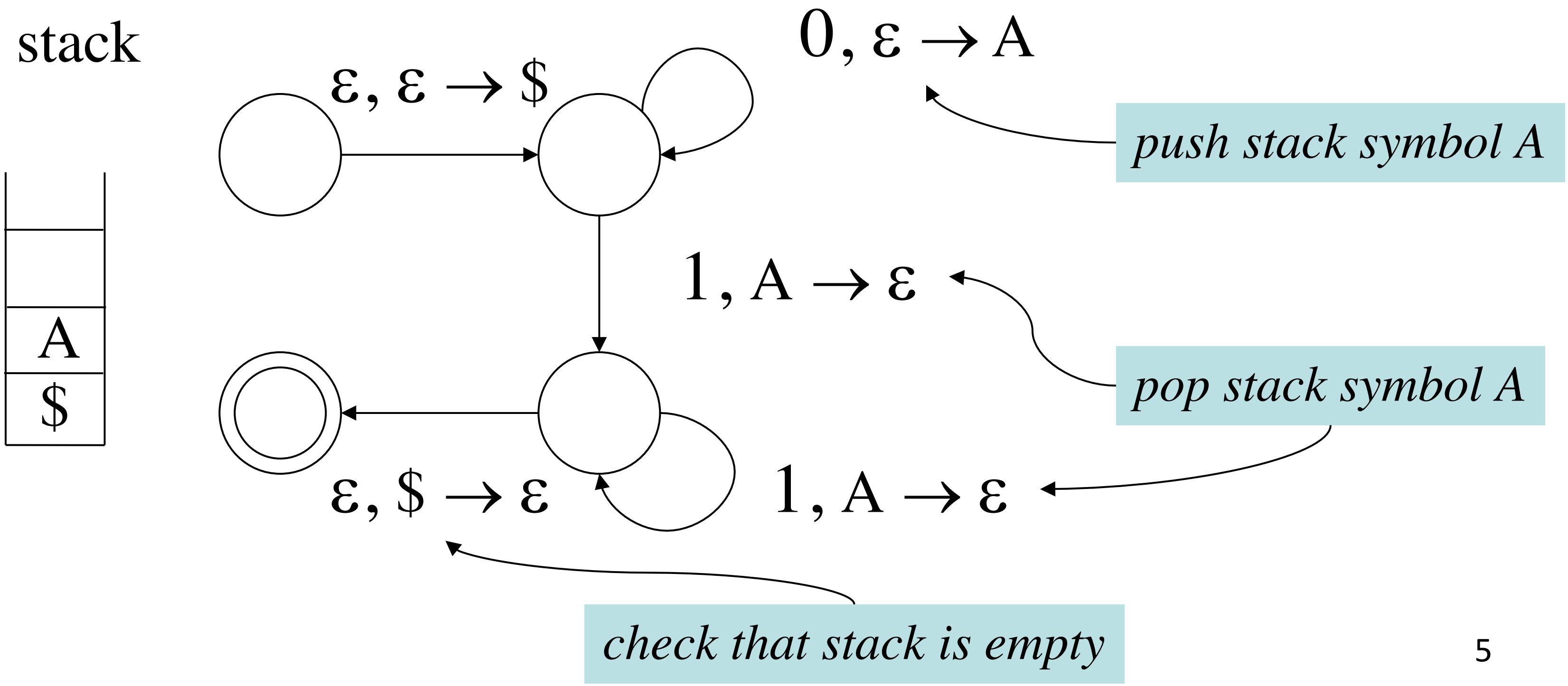
- Similar to FSAs there are non-deterministic pda and deterministic pda
- Unlike in the case of FSAs we cannot always convert a npda to a dpda
- Our goal in compiler design will be to choose grammars carefully so that we can always provide a dpda for it
- Similar to the FSA case, a DFA construction provides us with the algorithm for lexical analysis,
- In this case the construction of a dpda will provide us with the algorithm for parsing (take in strings and provide the parse tree)
- We will study later how to convert a given CFG into a parser by first converting into a PDA

Pushdown Automata

- PDA has
  - an alphabet (terminals),
  - stack symbols (like non-terminals and terminals),
  - a finite-state automaton,
  - stack

e.g. PDA for language
$L = \{ 0^n1^n : n >= 1 \}$

→ implies a push/pop of stack symbol(s)



$\varepsilon, \varepsilon \rightarrow \$$

$0, \varepsilon \rightarrow A$

push stack symbol A

$1, A \rightarrow \varepsilon$

pop stack symbol A

$1, A \rightarrow \varepsilon$

$\varepsilon, \$ \rightarrow \varepsilon$

check that stack is empty

Non-CF Languages

$$L_1 = \{wcw \mid w \in (a|b)*\}$$

$$L_2 = \{a^n b^m c^n d^m \mid n \geq 1, m \geq 1\}$$

$$L_3 = \{a^n b^n c^n \mid n \geq 0\}$$

CF Languages

$$L_4 = \{wcw^R \mid w \in (a|b)*\}$$

$$S \rightarrow aSa \mid bSb \mid c$$

$$L_5 = \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

$$S \rightarrow aSd \mid aAd$$

$$A \rightarrow bAc \mid bc$$

Summary

- CFGs can be used describe PL
- Derivations correspond to parse trees
- Parse trees represent structure of programs
- Ambiguous CFGs exist
- Some forms of ambiguity can be fixed by changing the grammar
- CF languages can be recognized using Pushdown Automata