CS312 Notes

October 13, 2024

Contents

1 Pushdown Automata

A pushdown automata (or PDA) is similar to an NFA but it has a stack. The stack provides additional memory beyond finite memory available in control; it allows the PDA to recognize some nonregular languages.

2 options to prove htat a language is context-free

- Construct a CFG that generates it
- Construct a PDA that recognizes it

Some CFLs are more easily descibed in terms of their generators, whereas others are more easily described in terms of their recognizers. Let's draw a schematic representation of the difference between an NFA and a PDA.

1.1 Terminology

- Writing a symbol on the stack is called pushing the symbol
- Removing a symbol from the stack is called popping the symbol
- All access to the stack may be done only at the top (LIFO storage device)

The primary benefit of that stack is that it can hold an **unlimited** amount of data; a PDA can recognize $\{0^n1^n \mid n \geq 0\}$ because it can use the stack to remember that number of 0s it has seen (read)

1.2 Informal Algorithm for $\{0^n1^n \mid n \geq 0\}$

1. Read symbols from the input, push a 0 for each 0 you see
2. As soon as a 1 is read, pop a 0 off the stack (for each 1 read).
3. If input finishes when the stack become empty, accept; if stack becomes empty while there is still input or input finishes while the stack is not empty, reject.

A PDA may be nondeterminisitic. Languages as the one above do not require nondeterminism. However, the language $\{ww^R \mid w \in \{0,1\}^*\}$ would require nondeterminism. Why?

1.3 Formalization

- A PDA may different alphabets for input (Σ) and stack (Γ)
- Nondeterminism allows for the PDA to make transitions on empty input. Define $\Sigma_\epsilon = \Sigma \ \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \ \ \{\epsilon\}$
- The domain of the PDA transition function is Q x Σ_ϵ x Γ_ϵ, where Q is the set of states
- The range of the PDA transition funciton is P(Q x Γ_ϵ).

$$\delta : Qx\Sigma_\epsilon x\Gamma_\epsilon \to P(Qx\Gamma_\epsilon)$$

1.4 Formal Def

A PDA is a 6-tuple (Q, Σ, Γ, δ, q_0, F), where Q, Σ, Γ are finite sets of states

1.5 PDA Computation

A PDA M = (Q,Σ,Γ,δ,q_0,F) computes as follows...
M inputs w = $w_1w_2...w_m$, where each $w_i \in \Sigma_\epsilon$

1. $r_0 = q_0$,$s_0 = \epsilon$; begin with state state and empty stack
2. $(r_{i+1},b) \in \delta(r_i,w_{i+1},a)$, i = 0,1,...m-1, where s_i = at and s_{i+1} = bt for some a,b $\in \Gamma$ and t $\in \Gamma^*$
3. $r_m \in F$; accept state encountered at end of input

1.6 Stack Notation

a,b $\rightarrow$ or simply abc

- a = input
- b = what are you popping
- c = push onto stack

A is read from the input, b is poppped from the stack, and c is pushed onto the stack
ϵ-cases

- If a = ϵ, machine can transition without reading any input
- if b = ϵ, machine can transition without popping any symbol from the stack
- if c = ϵ, machine can transition without writing any symbol onto the stack

read pop push

1.7 Empty Stack

The PDA does not consider the testing of an empty stack. We can achieve this by initially placing a special char (say)$onthestack.WhenthePDAencounterthatchar()$ again (on the stack), it knows the stack is effectively empty.

Both CFGs and PDAs specify context-free languages; we can always convert a CFG into a PDA that recognizes the language of the CFG

1.8 Theorem

CFG - specifies a program language
PDA - specifies/implements the compiler

A language is context-free $\iff$ $somePDArecognizesit$

1.9 Difficulties

How do we decide which substitutions to make for a derivation? (PDA P nondeterminism can help)

- At each step of the derivation one of the rules for a particular variable is selected non-deterministically
- P has to start by writing the start variable on the stack and then continue working the string w
- If while consuming the string w, P arrives at a string of terminals that equals w

1.10 Informal Description

Place marker symbol $ and tart varaible on the stack
Repeat:

- If TOS is a variable symbol A, non-deterministically select a rule r such that lhs(r) = A and substitute A by the string rhs(s)
- If TOS is a terminal symbol, a, read the next input symbol and compare it with a; if they match, pop the stack; if they do not match, reject this branch of nondeterminism
- If TOS is a $ and all the text has been read, accept; otherwise reject

1.11 Generic State Diagram

1. TOS = variable: set $\delta(q_{loop}, \epsilon, A) = \{(q_{loop}, w) \mid A \to w \in R\}$, where R is the set of CFG rules

2. TOS = terminal: set $\delta(q_{loop}, a, a\}) = \{(q_{loop}, \epsilon\})$

3. TOS = \$: $\delta(q_{loop}, \epsilon, \$) = \{(q_{accept}, \epsilon)\}$