BLG311E Formal Languages and Automata

Pushdown Automata(PDA) and Recognizing Context-free Languages

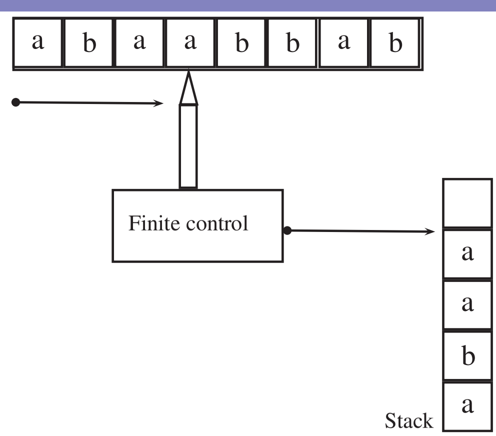
A.Emre Harmancı Osman Kaan Erol Tolga Ovatman

2012

It is not possible to design finite automata for every context-free language. For instance the recognizer for the language $\omega\omega^R || omega \in \Sigma^*$ should contain a memory. We can design a pushdown automaton for every context-free language.

Pushdown Automata

A pushdown automaton is similar in some respects to a finite automaton but has an auxiliary memory that operates according to the rules of a stack. The default mode in a pushdown automaton (PDA) is to allow nondeter- minism, and unlike the case of finite automata, the nondeterminism cannot always be eliminated.



PDAs are not deterministic. Input strip is only used to read input while the stack can be written and read from.

Formal Definition of a PDA

A pushdown automaton (PDA) is a 6-tuple $M = (S, \Sigma, \Gamma, \Delta, s_0, F)$, where:

- S: A finite, non-empty set of states where $s \in S$.
- Σ: Input alphabet (a finite, non-empty set of symbols)
- Γ: Stack alphabet
- $s_0\ inS$: An initial state, an element of S.
- δ: The state-transition relation $\delta \subseteq (S \times \Sigma^* \times \Gamma^*) \times (S \times \Gamma^*)$
- F: The set of final states where $F \subseteq S$.

An example

$(\omega \subset \omega^R | \omega \in \{a,b\}^*)$
$M = (S, \Sigma, \Gamma, \Delta, s_0, F)$
$S = \{s_0, f\}$, $\Sigma = \{a,b,c\}$, $\Gamma = \{a,b\}$, $F = \{f\}$
$\Delta = \{[(s,a,\Lambda),(s,a)],[(s,b,\Lambda),(s,b)],[(s,c,\Lambda),(f,\Lambda)],$
$[(f,a,a),(f,\Lambda)],[(f,b,b),(f,\Lambda)]\}$

state	tape	stack	trans. rule
s	abb **c** bba	Λ	[(s,a,Λ),(s,a)]
s	bb **c** bba	a	[(s,b,Λ),(s,b)]
s	b **c** bba	ba	[(s,b,Λ),(s,b)]
s	**c** bba	bba	[(s,c,Λ),(f,Λ)]
f	bba	bba	[(f,b,b),(f,Λ)]
f	ba	ba	[(f,b,b),(f,Λ)]
f	a	a	[(f,a,a),(f,Λ)]
f	Λ	Λ	

An example

state	tape	stack	trans. rule
s	abb **c** bba	Λ	[(s,a,Λ),(s,a)]
s	bb **c** bba	a	[(s,b,Λ),(s,b)]
s	b **c** bba	ba	[(s,b,Λ),(s,b)]
s	**c** bba	bba	[(s,c,Λ),(f,Λ)]
f	bba	bba	[(f,b,b),(f,Λ)]
f	ba	ba	[(f,b,b),(f,Λ)]
f	a	a	[(f,a,a),(f,Λ)]
f	Λ	Λ	

$G = (N, \Sigma, n_0, \mapsto)$
$N = \{S\}$
$\Sigma = \{a,b,c\}$
$n_0 = S$
$<S>::= a<S>a \mid b<S>b|c$

Definitions

Push: To add a symbol to the stack $[(p,u,\Lambda),(q,a)]$

Pop: To remove a symbol from the stack $[(p,u,a),(q,a)]$

Configuration: An element of $S \times \Sigma^* \times \Gamma^*$. For instance (q,xyz,abc) where a is the top of the stack, c is the bottom of the stack.
Instantaneous description (to yield in one step):

Let $[(p,u,\beta),(q,\gamma)] \in \Delta$ and $\forall x \in \Sigma^* \wedge \forall \alpha \in \Gamma^*$

$(p,ux,\beta\alpha) \vdash_M (q,x,\gamma\alpha)$
Here u is read from the input tape and β is read from the stack while γ is written to the stack.

Definitions

$(p,ux,\beta\alpha) \vdash_M (q,x,\gamma\alpha)$

Let $\vdash_M{}^*$ be the reflexive transitive closure of $\vdash_M$ and let $\omega \in \Sigma^*$ and s_0 be the initial state. For M automaton to accept ω string:

$(s,\omega,\Lambda) \vdash_M{}^* (p,\Lambda,\Lambda)$ and $p \in F$
$C_0 = (s,\omega,\Lambda)$ and $C_n = (p,k,\Lambda)$ where
$C_0 \vdash_M C_1 \vdash_M \ldots \vdash_M C_{n-1} \vdash_M C_n$

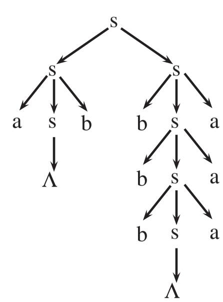
This operation is called *computation* of automaton M, this computation invloves n steps.

Let $L(M)$ be the set of string accepted by M.
$L(M) = \{\omega | (s,\omega,\Lambda) \vdash_M{}^* (p,\Lambda,\Lambda) \wedge p \in F\}$

Example 1

$\omega \in \{\{a,b\}^* | \#(a) = \#(b)\}$
$M = (S,\Sigma,\Gamma,\Delta,s_0,F)$
$\Delta = \{[(s,\Lambda,\Lambda),(q,c)],[(q,a,c),(q,ac)],[(q,a,a),(q,aa)],$
$[(q,a,b),(q,\Lambda)],[(q,b,c),(q,bc)],[(q,b,b),(q,bb)],$
$[(q,b,a),(q,\Lambda)],[(q,\Lambda,c),(f,\Lambda)]\}$

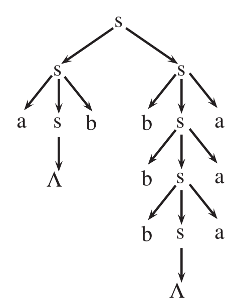
state	tape	stack	trans. rule
s	abbbabaa	Λ	[(s,Λ,Λ),(q,c)]
q	abbbabaa	c	[(q,a,c),(q,ac)]
q	bbbabaa	ac	[(q,b,a),(q,Λ)]
q	bbabaa	c	[(q,b,c),(q,bc)]
q	babaa	bc	[(q,b,b),(q,bb)]
q	abaa	bbc	[(q,a,b),(q,Λ)]
q	baa	bc	[(q,b,b),(q,bb)]
q	aa	bbc	[(q,a,b),(q,Λ)]
q	a	bc	[(q,a,b),(q,Λ)]
q	Λ	c	[(q,Λ,c),(f,Λ)]
f	Λ	Λ	



Example 1

$\omega \in \{\{a,b\}^* | \#(a) = \#(b)\}$
$M = (S,\Sigma,\Gamma,\Delta,s_0,F)$
$\Delta = \{[(s,\Lambda,\Lambda),(q,c)],[(q,a,c),(q,ac)],[(q,a,a),(q,aa)],$
$[(q,a,b),(q,\Lambda)],[(q,b,c),(q,bc)],[(q,b,b),(q,bb)],$
$[(q,b,a),(q,\Lambda)],[(q,\Lambda,c),(f,\Lambda)]\}$

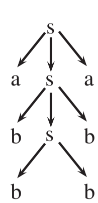
$G = (N,\Sigma,n_0,\mapsto)$
$N = \{s\}$
$\Sigma = \{a,b\}$
$n_0 = s$
$<s> ::= a <s> b \mid b <S> a \mid\ <s><s>\ \mid \Lambda$



Example 2

$\omega \in \{xx^R | x \in \{a,b\}^*\}$
$M = (S,\Sigma,\Gamma,\Delta,s_0,F)$
$\Delta =$
$\{[(s,a,\Lambda),(s,a)],[(s,b,\Lambda),(s,b)],[(s,\Lambda,\Lambda),(f,\Lambda)],[(f,a,a),(f,\Lambda)],[(f,b,b),(f,\Lambda)]\}$

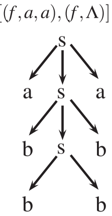
state	tape	stack	trans. rule
s	abbbba	Λ	[(s,a,Λ),(s,a)]
s	bbbba	a	[(s,b,Λ),(s,b)]
s	bbba	ba	[(s,b,Λ),(s,b)]
s	bba	bba	[(s,Λ,Λ),(f,Λ)]
f	bba	bba	[(s,Λ,Λ),(f,Λ)]
f	bba	bba	[(f,b,b),(f,Λ)]
f	ba	ba	[(f,b,b),(f,Λ)]
f	a	a	[(f,a,a),(f,Λ)]
f	Λ	Λ	



Example 2

$\omega \in \{xx^R | x \in \{a,b\}^*\}$
$M = (S,\Sigma,\Gamma,\Delta,s_0,F)$
$\Delta =$
$\{[(s,a,\Lambda),(s,a)],[(s,b,\Lambda),(s,b)],[(s,\Lambda,\Lambda),(f,\Lambda)],[(f,a,a),(f,\Lambda)],[(f,b,b),(f,\Lambda)]\}$

$G = (N,\Sigma,n_0,\mapsto)$
$N = \{s\}$
$\Sigma = \{a,b\}$
$<s> ::= a <s> a \mid b <S> b \mid aa \mid bb$



Deterministic PDA

Deterministic PDA

1) $\forall s \in S \wedge \forall \gamma \in \Gamma$ if $\delta(s,\Lambda,\gamma) \neq \varnothing \Rightarrow \delta(s,\sigma,\gamma) = \varnothing; \forall \sigma \in \Sigma$
2) If $a \in \Sigma \cup \{\Lambda\}$ then $\forall s, \forall \gamma$ and $\forall a$ Card$(\delta(s,a,\gamma)) \leq 1$

- (1) If there exists a lambda transition(yielding in one step) in a configuration no other transitions should be present for any other input. (2) There should be a unique transition for any (state,symbol,stack symbol) tuple
- For nondeterministic PDA, the equivalence problem to deterministic PDA is proven to be undecidable[1].
- For instance $\omega\omega^R$ can be accepted by a non-deteministic PDA but there doesn't exist any deterministic PDA that accepts this language.

[1] An undecidable problem is a decision problem for which it is impossible to construct a single algorithm that always leads to a correct yes-or-no answer