ECS120 Introduction to the Theory of Computation
Fall Quarter 2007

Discussion Notes
Wednesday, November 7, 2007

Context-Free Languages

Pushdown Automata

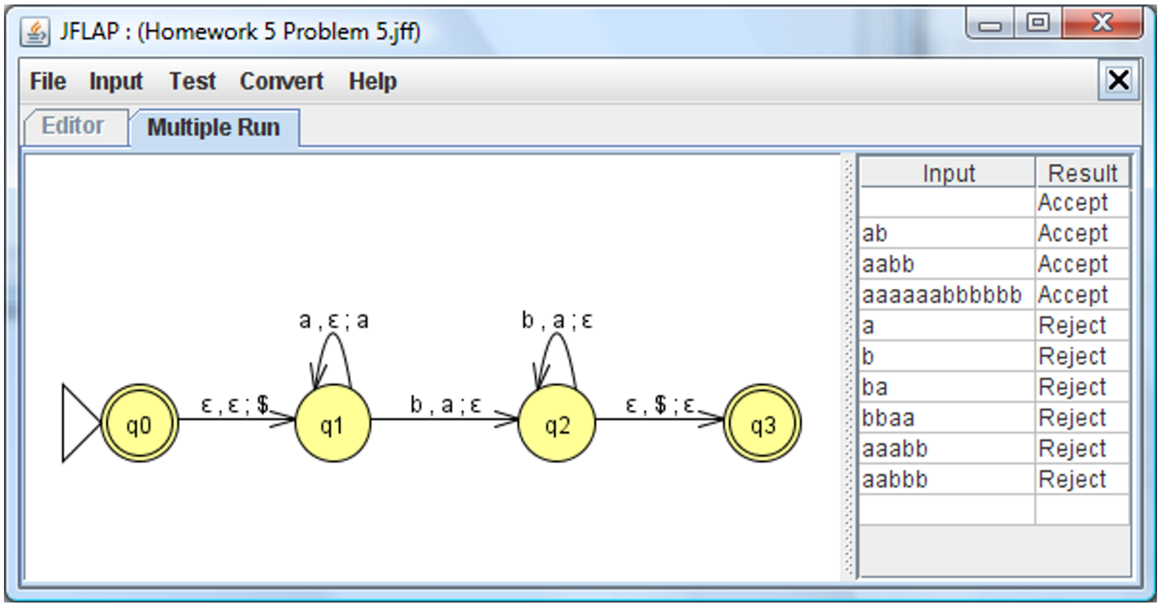
A pushdown automaton is a 6-tuple $P = (\,Q, \Sigma, \Gamma, \delta, q_0, F\,)$ where:

1. Q is the finite set of states
2. Σ is the finite input alphabet
3. Γ is the finite stack alphabet
4. $q_0 \in Q$ is the start state
5. $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ is the transition function
6. $F \subseteq Q$ is the set of accept states

Computation in a pushdown automaton is a bit more complicated than in a DFA or NFA. The PDA P accepts input $w = w_1 w_2 \cdots w_m$, where $w_i \in \Sigma_\epsilon$, if a sequence of states $r_0, r_1, \ldots, r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ exist such that:

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

Consider the PDA from last week for the language $L = \{\, a^n b^n \mid n \geq 0 \,\}$:



Input	Result
	Accept
ab	Accept
aabb	Accept
aaaaaabbbbbb	Accept
a	Reject
b	Reject
ba	Reject
bbaa	Reject
aaabb	Reject
aabbb	Reject

Let's work an example to see the computation in progress. Consider the word *aabb*.

	at	input	pop	next	push	stack
0				q_0		ϵ
1	q_0	ϵ	ϵ	q_1	\$	\$
2	q_1	a	ϵ	q_1	a	$a\$$
3	q_1	a	ϵ	q_1	a	$aa\$$
4	q_1	b	a	q_2	ϵ	$a\$$
5	q_2	b	a	q_2	ϵ	\$
6	q_2	ϵ	\$	q_3	ϵ	ϵ
$\uparrow$		$\uparrow$		$\uparrow$		$\uparrow$
i		w		r		s

Therefore we have:

i	0	1	2	3	4	5	6
w	$-$	ϵ	a	a	b	b	ϵ
r	q_0	q_1	q_1	q_1	q_2	q_2	q_3
s	ϵ	\$	$a\$$	$aa\$$	$a\$$	\$	ϵ

which satisfies our requirements.

Closure Properties

Context-free languages are closed under union, concatenation, and star.

You were asked to prove CFLs are closed under the star operation for homework 5. (The proof is in the homework solutions.)

Let $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$ be context-free grammars.

Let $G_3 = (V_3, \Sigma, R_3, S_3)$ be a CFG such that $L(G_3) = L(G_1) \cup L(G_2)$ where:

- $V_3 = V_1 \cup V_2 \cup S_3$
- $R_3 = R_1 \cup R_2 \cup (S_3 \to S_1 \mid S_2)$

Let $G_4 = (V_4, \Sigma, R_4, S_4)$ be a CFG such that $L(G_4) = L(G_1) \circ L(G_2)$ where:

- $V_4 = V_1 \cup V_2 \cup S_4$
- $R_4 = R_1 \cup R_2 \cup (S_4 \to S_1\, S_2)$

However, CFLs are not closed under complementation and intersection.

Example: PDA $\cap$ DFA

Let C be a context-free language and R be a regular language. Prove that the language $C \cap R$ is context free. (Sipser 2.18)

This problem is answered on page 133 of your book.

Let $P = (\, Q_p, \Sigma, \Gamma, \delta_p, p_0, F_p \,)$ be a PDA such that $L(P) = C$ and $D = (\, Q_d, \Sigma, \delta_d, d_0, F_d \,)$ a DFA such that $L(D) = R$.

Let $P' = (\, Q, \Sigma, \Gamma, \delta, q_0, F \,)$ be a PDA such that $L(P') = C \cap R$ where:

- $Q = Q_p \times Q_d$
- $q \in F$ if and only if $q \in F_p \times F_d$
- $q_0 = (p_0, d_0)$

The transition function δ will do what P does *and* keep track of the states of D. More formally, let $p_i, p_j \in Q_p$, $d_i \in Q_d$, $\alpha \in \Sigma - \epsilon$, and $\beta, \gamma \in \Gamma$.

1. For every transition in δ_p where:

$$(p_i, \alpha, \beta) \rightarrow (p_j, \gamma)$$

   Add the following transition to δ for every state $d_i \in Q_d$:

$$((p_i, d_i), \alpha, \beta) \rightarrow ((p_j, \delta_d(d_i, \alpha)), \gamma)$$

2. For every transition in δ_p where:

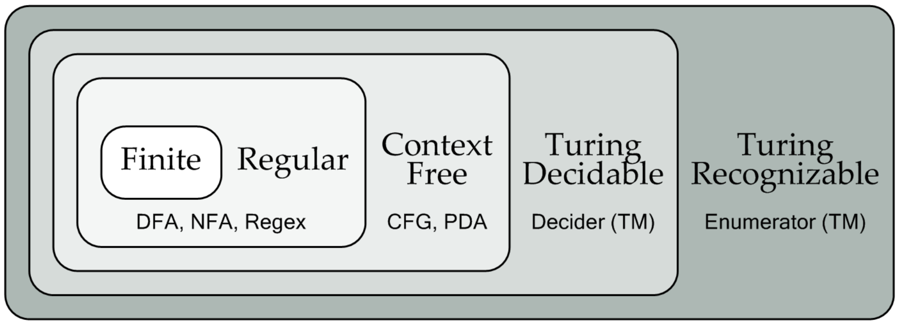
$$(p_i, \epsilon, \beta) \rightarrow (p_j, \gamma)$$

   Add the following transition to δ for every state $d_i \in Q_d$:

$$((p_i, d_i), \epsilon, \beta) \rightarrow ((p_j, d_i), \gamma)$$

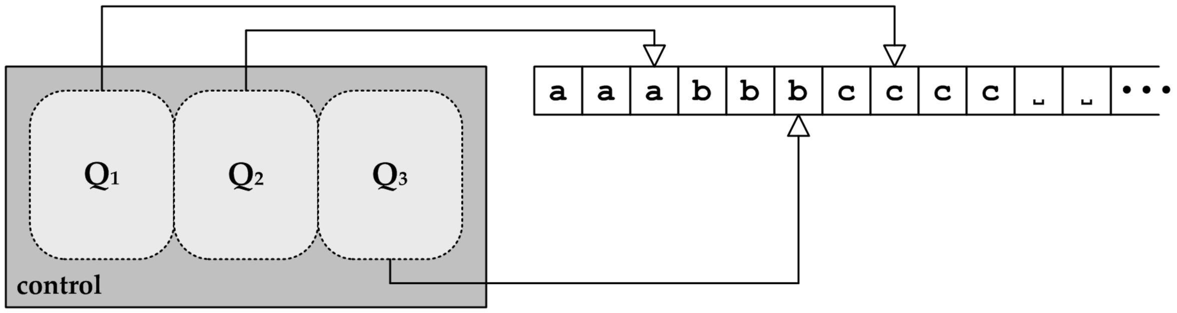
Language Hierarchy

With the introduction of Turing Machines, we have now seen several different types of languages. In general, there are the languages which are accepted by finite automata, pushdown automata, and turning machines. More specifically we have:



Turning Machine Variants

Consider a **Multi-Head Turing Machine**. This is a single-tape Turing machine with multiple read/write heads. However, in every state only one head may be used. Therefore, for k heads we have a partition of states $Q_1, Q_2, \ldots, Q_k$ where each Q_i contains the set of states which use the ith head. For $k = 3$, we might have something like:



Equivalence

A multi-head Turing machine is equivalent to a normal Turing machine. Let M_{MH} be a multi-head Turing machine and M_{TM} be a normal Turing machine. To show that they are equivalent, we must provide two arguments.

Argument 1: For any normal Turing machine M_{TM}, there exists some multi-head Turing machine M_{MH} such that $L(M_{\text{TM}}) = L(M_{\text{MH}})$.

To show this, we need to show that M_{TM} can be simulated by some M_{MH}. This part of the proof is simple. We can build M_{MH} such that it forces all the states to use the same read/write head. The rest of the heads can be ignored. The states and transitions remain the same as in M_{TM}.

Argument 2: For any multi-head Turing machine M_{MH}, there exists some normal Turing machine M_{TM} such that $L(M_{\text{MH}}) = L(M_{\text{TM}})$.

We must show that M_{MH} can be simulated by some M_{TM}. This part is trickier. We could do something similar to the proof for a multi-tape Turing machine, and use some modifier on the symbols to track where the different heads are located on the tape. For example:

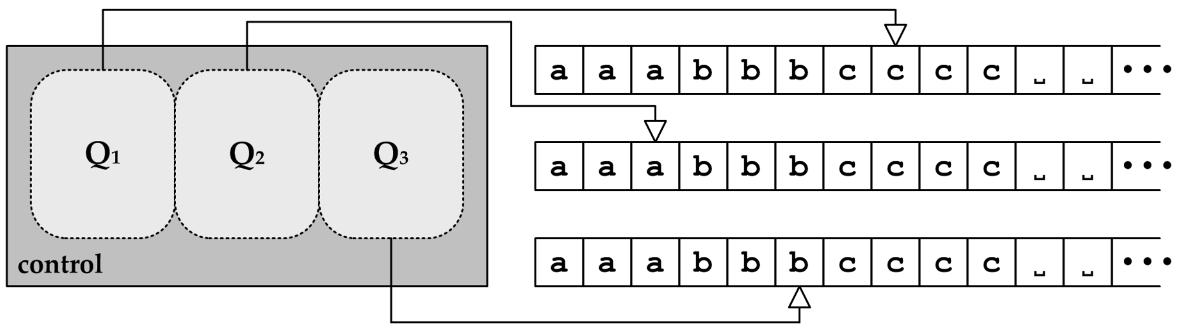
a	a	ȧ	b	b	ḅ	c	c	c	c’	␣	···

However, we already know multi-tape Turing machines are equivalent to normal Turing machines. We could alternatively show that for any multi-head Turing machine M_{MH}, there exists some multi-tape Turing machine M_{MT} such that $L(M_{\text{MH}}) = L(M_{\text{MT}})$.

Lets create a k-tape Turing machine M_{MT} to simulate a k-head Turing machine M_{MH} as follows:

1. Initialize every tape in M_{MT} with the input string.
2. Whenever head i is moved in M_{MH}, move the head in the ith tape in M_{MT}.
3. Whenever any head in M_{MH} writes a symbol, write the same symbol on the same position on every tape in M_{MT}.

Everything else would be the same in M_{MT} as in M_{MH}. So we would end up with something like:


Q1
Q2
Q3
control
a a a b b b c c c c ␣ ␣ •••
a a a b b b c c c c ␣ ␣ •••
a a a b b b c c c c ␣ ␣ •••

Since we know $L(M_{\text{MT}}) = L(M_{\text{TM}})$, we can conclude $L(M_{\text{MH}}) = L(M_{\text{MT}}) = L(M_{\text{TM}})$. Of course, all we have given here is an abstract idea of how to do this. The devil is in the details. You need to convince yourself that every step is correct.