cs302 Theory of Computation UVa Spring 2008

Notes: Pushdown Automata

Tuesday, 5 February

Upcoming Schedule

Wednesday, 6 February (9:30-10:30am): Theory Coffee Hours (Wilsdorf Coffee Shop, I may be at one of the tables upstairs)
Wednesday, 6 February (6-7pm): Problem-Solving Session (Olsson 226D)
Thursday, 7 February: Problem Set 2 is due **at the beginning of class**.

Proving Non-Regularity

Pumping Lemma. If A is a regular language, then there is a number p (the pumping length) where for any string $s \in A$ and $|s| \geq p$, s may be divided into three pieces, $s = xyz$, such that $|y| > 0$, $|xy| \leq p$, and for any $i \geq 0$, $xy^iz \in A$.

To use the pumping lemma to prove a language A is non-regular, assume A is regular and find a contradiction using the pumping lemma. Since the pumping lemma says that the property holds for *any* string $s \in A$, if we can find *one* string $w \in A$ for which the property does not hold (that is, we need to show there is *no way* to divide w into xyz with the necessary properties) then we have our contradiction.

Example 1. Prove the language $\left\{0^i1^j | i \leq j\right\}$ is not regular.

Proof by Contradiction.
Assume $\left\{0^i1^j | i \leq j\right\}$ is regular and p is the pumping length for A. Then, we will identify a string that cannot be pumped.

Choose $w = 0^p1^p$. $w \in A$ since we can choose $i = j = p$.
The pumping lemma says that $w = xyz$ for some x, y, and z such that $xy^iz \in A$ for all $i \geq 0$, and $|xy| \leq p$.
Since the first p symbols in w are 0s, no matter how we choose x, y, and z, we know $|xy| \leq p$, so y must be withing the first p symbols of w, hence it can only contain 0s.
But, since y only includes 0s, pumping y increases the number of 0s, without changing the number of 1s. To be in the language, though, the number of 0s (i) must be $\geq$ the number of 1s (j).
Thus, we have a contradiction. This proves that the language is not regular.

Example 2. Prove the language $\{ww | w \in \Sigma^*\}$ is not regular.

Example 3. Prove the language $\left\{w | w \in \{0,1\}^* \text{ and the number of 0s in w exceeds the number of 1s}\right\}$ is not regular.

Pushdown Automata

A *pushdown automata* is a finite automaton with a stack. A stack is a data structure that can contain any number of elements, but for which only the top element may be accessed. We can represent a stack as a sequence of elements, $[s_0, s_1, \ldots, s_n]$. We use Γ (Gamma) to represent the stack alphabet. Γ is a finite set of symbols. So, a stack is represented by Γ^*.

There are two operations on a stack:

> *push*: $\Gamma^* \times \Gamma_\epsilon \rightarrow \Gamma^*$. *push* is defined by:
>
> > $push(s, \epsilon) = s$
> > for $v \in \Gamma, s = [s_0, \ldots, s_n], (s, v) = [v, s_0, s_1, \ldots, s_n]$
>
> *pop*: $\Gamma^* \rightarrow \Gamma^* \times \Gamma_\epsilon$. *pop* is defined by:
>
> > $pop([]) = ([], \epsilon)$
> > $pop([s_0, \ldots, s_n]) = (s_0, [s_1, \ldots, s_n])$

A *deterministic pushdown automaton*[1] (DPDA) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q, Σ, q_0, and F are defined as they are for a deterministic finite automaton, Γ is a finite state (the stack alphabet), and

> δ: $Q \times \Sigma \times \Gamma_\epsilon \rightarrow Q \times \Gamma_\epsilon$

We can use any symbols we want in the stack alphabet, Γ. As with state labels, in designing a DPDA, it is important to give symbols names that have meaning. Typically, we use $ as a special symbol, often meaning the bottom of the stack.

We use label arrows in a DPDA as $\Sigma, \Gamma_\epsilon \rightarrow \Gamma_\epsilon$. For $a \in \Sigma$, $b, c \in \Gamma$:

- $a, b \rightarrow c$ means if the current input is a and the top-of-stack is b, follow this transition and pop the b off the stack, and push the c.
- $a, \epsilon \rightarrow c$ means if the current input is a, follow this transition and push c on the stack. (It doesn't matter what is on the stack.)
- $a, b \rightarrow \epsilon$ means if the current input is a and the top-of-stack is b, follow this transition and pop the b off the stack.
- $a, \epsilon \rightarrow \epsilon$ means if the current input is a, follow this transition (and don't modify the stack).

Here is an example DPDA - what language does it recognize?

Prove that a DPDA is more powerful than a DFA.

[1] Note that the book (Definition 2.1) defines a *nondeterministic pushdown automaton*, but does not define a *deterministic pushdown automaton*.