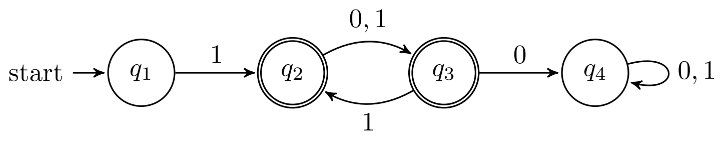
CS 525: Theory of Computation
Midterm 1

Dustin Ingram

January 31, 2012

1. **Solution:**



2. **Solution:** Using the pumping lemma, assume L_2 is regular. Let $s = b^{p+1}a^p$, which satisfies the condition that $|b| > |a|$. Then, if s is split into xyz to satisfying the conditions of the pumping lemma, by the 3^{rd} condition of the pumping lemma, y consists only of b's. Now, consider when $xy^0z = xz$. This decreases the number of b's in s such that $|b| = |a|$, breaking the condition that $|b| > |a|$, proving that s cannot be a member of L_2, thereby proving that L_2 is not regular.

3. **Solution:** Using the non-determinism of the PDA, the machine has two separate branches based on whether it's initial character is a or b. Before the first character is read, the PDA pushes a \$ symbol onto the stack (to later test for an empty stack). If the first character in the string is an a, the PDA first reads and pushes all the a's. Once it encounters a b, the PDA pops an a off the stack. The PDA continues, pushing every a and popping an a every time a b is read. The PDA completes successfully if it reaches the end of the string and the stack is empty (using the \$ symbol procedure). If at any point it reaches the end of the string and the stack is not empty, the PDA has failed (the end of the stack, however, may be repeatedly reached).

4. **Solution:** Here, we will use the pumping lemma for context-free languages. Let $s = w\#t$ (instead of $u\#v$), where $|w| = p$ and thus since $|w| = |t|$, $|t| = p$ as well. In this case, vxy must be either:

   (a) Wholly contained within the substring w;

   (b) Wholly contained within the substring t;

(c) Split between w and t, thus somehow containing the '#' symbol.

In 1), pumping v and y would result in some modification of the substring w, without modification to the substring t, therefore not guaranteeing that w is a permutation of t. In 2), the same is true as in 1), with w and t interchanged. In 3), vxy must contain the '#' symbol, which must be in x (otherwise pumping would produce more than one). Again, in this case, pumping v and y would not guarantee that the substring w is still a permutation of t. For example, if we take $s = aaab\#aaba$, where $p = 4$, we have two possible divisions of s; in the case where we pump v and y once such that $uv^1xy^1z => uv^2xy^2z$, the following is produced:

$$\overbrace{\underbrace{aa}_{u}\,\underbrace{ab}_{v}}^{w}\,\underbrace{\#}_{x}\,\overbrace{\underbrace{a}_{y}\,\underbrace{aba}_{z}}^{t} \rightarrow \overbrace{\underbrace{aa}_{u}\,\underbrace{abab}_{v^2}}^{w}\,\underbrace{\#}_{x}\,\overbrace{\underbrace{aa}_{y^2}\,\underbrace{aba}_{z}}^{t}$$

$$\overbrace{\underbrace{aaa}_{u}\,\underbrace{b}_{v}}^{w}\,\underbrace{\#}_{x}\,\overbrace{\underbrace{aa}_{y}\,\underbrace{ba}_{z}}^{t} \rightarrow \overbrace{\underbrace{aaa}_{u}\,\underbrace{bb}_{v^2}}^{w}\,\underbrace{\#}_{x}\,\overbrace{\underbrace{aaaa}_{y^2}\,\underbrace{ba}_{z}}^{t}$$

Here it is clear that pumping s does not produce a string in L_4, and thus L_4 is not context-free.

5. **Solution:** Let M_5 be a Turing machine that accepts the language of a string s of 0's whose length is a prime number. Here, M_1 will use two special characters '\$' and '#'. The machine will essentially check whether the string contains a multiple of $n \in \{2, 3, 4, 5, \ldots, |s|\}$. To begin, the machine marks the first character in the string as a '\$'. The length of this string of '\$' characters tells the machine how many 0's to skip before it marks the next character as a '#'; in this case the length is 1 (and thus we are checking to see if it is a multiple of 2). The machine then returns to the beginning of the string, marking every other character as a '#'. If the machine reaches the end of the string and the last character is marked as a '#', it is not prime and thus the machine fails. However, if the machine reaches the end of the string and the last character cannot be marked as '#', the machine returns to the beginning of the string, writing a new '\$' character after the last '\$' character, thus incrementing the factor to test. The machine terminates successfully if it marks the entire string as '\$' characters (thus, the last character is a '\$' and not a '#' or a '0').

6. **Solution:** A deterministic Turing machine must have only three states: an accept state, a reject state, and a state which is neither accept or reject. To simulate any deterministic Turing machine with more than three states, we simply must store the state transitions on the tape, such that for each possible transition, we show that it either transitions to the accepting state, the rejecting state, or remains in the 'processing' state. Thus, any deterministic Turing machine can be simulated.