

CIS 262 Fall 2009: Solutions to Homework 4

Problem 1

Exercise 6.2.5 from textbook.

Answer:

- a) $(q_0, bab, Z_0) \vdash (q_2, ab, BZ_0) \vdash (q_3, b, Z_0) \vdash (q_1, b, AZ_0) \vdash (q_1, \varepsilon, Z_0) \vdash (q_0, \varepsilon, Z_0) \vdash (f, \varepsilon, \varepsilon)$.
Since the final state is f , bab is accepted.
- b) $(q_0, abb, Z_0) \vdash (q_1, bb, AAZ_0) \vdash (q_1, b, AZ_0) \vdash (q_1, \varepsilon, Z_0) \vdash (q_0, \varepsilon, Z_0) \vdash (f, \varepsilon, \varepsilon)$.
Since the final state is f , abb is accepted.
- c) $(q_0, b^7a^4, Z_0) \vdash (q_2, b^6a^4, BZ_0) \vdash (q_2, b^5a^4, B^2Z_0) \vdash (q_2, b^4a^4, B^3Z_0) \vdash (q_2, b^3a^4, B^4Z_0) \vdash (q_2, b^2a^4, B^5Z_0) \vdash (q_2, ba^4, B^6Z_0) \vdash (q_2, a^4, B^7Z_0) \vdash (q_3, a^3, B^6Z_0) \vdash (q_2, a^3, B^5Z_0) \vdash (q_3, a^2, B^4Z_0) \vdash (q_2, a^2, B^3Z_0) \vdash (q_3, a, B^2Z_0) \vdash (q_2, a, BZ_0) \vdash (q_3, \varepsilon, Z_0) \vdash (q_1, \varepsilon, AZ_0)$. Hence the content of the stack is AZ_0 .
- d) $L(P) = \{w \mid \#(w, b) = 2\#(w, a)\}$.

Intuitively, the loop involving q_0 and q_1 takes care of a substring w' of w such that for any prefix w'' of w' , $\#(w'', b) \leq 2\#(w'', a)$ (we push 2 A 's for each a and pop one A for each b). The states q_2, q_3 take care of other substrings, since we push one B for each b and can pop 2 B 's for an a . But if we have a substring w' where $\#(w'', b) = 2\#(w'', a) - 1$, then from q_3 we cannot get back to q_2 , so we push an A in the stack and make a transition from q_3 to q_1 so that later we can match this additional A with a b to get $\#(w, b) = 2\#(w, a)$. Finally when all symbols are consumed and stack is empty, we make a transition to the final state f to accept the word in $L(P)$.

Problem 2

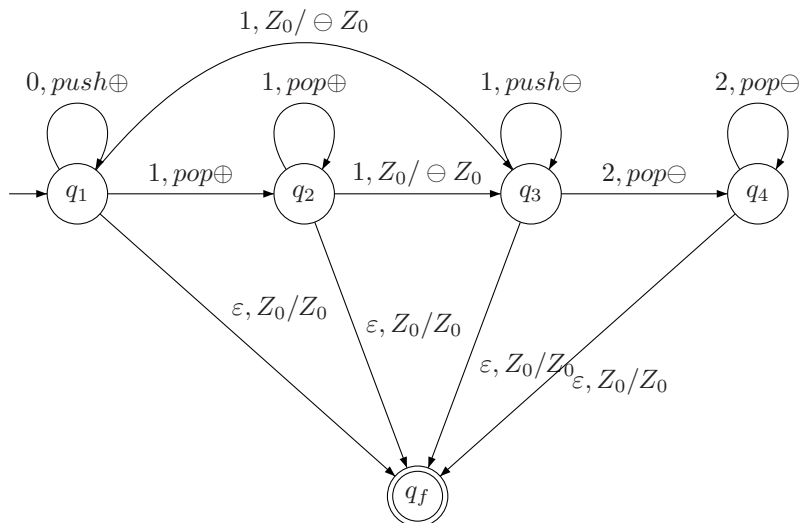
Draw a PDA P such that $L(P) = \{0^i1^j2^k \mid j = i + k\}$. Explain your construction.

Answer:

The PDA uses three stack symbols, \oplus, \ominus, Z_0 . Z_0 is the start symbol, and used to mark the bottom of the stack.

To recognize the string, we first push a \oplus on the stack for every 0, then we pop a \oplus off the stack for every 1 we see. If there are still more 1s we push a \ominus on the stack for each 1 we see after that. Finally we pop a \ominus for each 2 we see. A string is in the language exactly if all the pushes and pops canceled out and leaves us with the stack we started with, so we add an ε -transition to an accepting state whenever the top of the stack is Z_0 .

In the figure below $push\oplus$ is shorthand for $z/\oplus z$, and $pop\oplus$ is shorthand for \oplus/ε .



Problem 3

Let $\Sigma = \{a, b, \#\}$. Draw a PDA P such that $N(P) = \{u\#v \mid u, v \in \{a, b\}^*, u \neq v\}$. Explain your construction.

Answer: The PDA uses two stack symbols, Z and W . Z is the start stack symbol and marks the bottom of the stack, while W is used to count things.

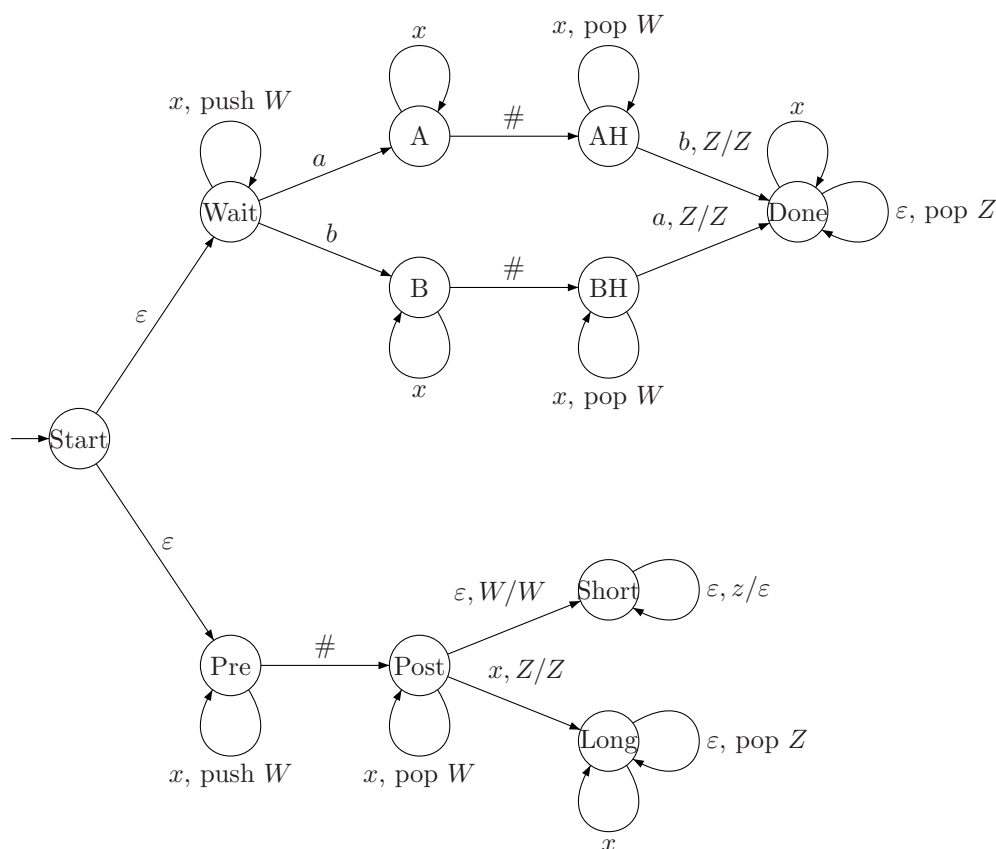
For a string of the form $u\#v$, $u \neq v$ if either (1) u and v have different lengths, or (2) the strings u and v have different symbols in position i , for some value of i . The PDA first nondeterministically guesses which of these two cases applies: for the first case it goes to the lower part of the diagram, and for the second case it goes to the upper part.

In the lower part, the PDA needs to check that the lengths before and after $\#$ must be different. It counts how long the string before the hash is by pushing W 's, and then verifies that the string after the hash is either longer or shorter by popping symbols.

In the upper part, it needs to check that the string u before the hash is of the form xx' and the string v after the hash is yy' such that x and y are of same length (or symmetrically, $u = xbx'$ and $v = yay'$ with $|x| = |y|$). It nondeterministically guesses where the mismatch will be, and pushes one W for each symbol until the mismatch on the stack. Thus, the height of stack corresponds to the length of x (the string before mismatch). It then remembers what the next letter is (by going to either state A or state B), ignore the rest of the string until it sees the hash symbol, counts the correct number of letters (by popping off all the W 's until only the Z remains), and verifies that the next letter is different (if it is not the PDA gets stuck). If it reaches Done we know there was a successful mismatch, and it accepts.

Note that in the upper part, the PDA simply skips over the symbols after the mismatch until the hash, and thus it is not checking (and does not need to check) that strings before and after hash are of same length, rather only that mismatched symbol in u is at same distance from beginning as the corresponding symbol in v from hash.

The figure below uses various shorthands: x means "either a or b " (but not $\#$); push W means z/Wz , pop W means W/ε , and an arc without any stack annotation means z/z (leave the stack unchanged).



Problem 4

Prove that the following language is not a CFL using the pumping lemma for CFLs: $\{ 0^m 1^n \mid n = m^2 \}$.

Answer:

We use the standard pumping-lemma for context-free languages. Assume L is context-free. Then there exists some number $p > 0$ s.t. for all $w \in L$ with $|w| \geq p$ we have $w = uvxyz$ where the following conditions hold: $|vxy| \leq p$; $|vy| > 0$; and $uv^i xy^i z \in L$ for all $i \geq 0$.

Let $w = 0^p 1^{p^2}$. Since $w \in L$ with $|w| > p$, the conditions specified by the pumping lemma must hold. However:

- if v (or y) contains both 0's and 1's, then $uv^2 xy^2 z \notin L$ since $uv^2 xy^2 z$ contains some 1's followed by 0's.
- if vxy contains only 0's (or only 1's) then $uv^2 xy^2 z \notin L$ since $N_1(uv^2 xy^2 z) \neq (N_0(uv^2 xy^2 z))^2$. Again we use $N_x(u)$ to denote the number of occurrences of x in u .

The only possibility left is that v contains some positive number of 0's and y contains some positive number of 1's. Therefore there exist numbers a, b, c, d such that $a, c > 0$ and $b, d \geq 0$ and $0^{a-i} 0^b 1^{c-i} 1^d \in L$ for all $i \geq 0$.

Inclusion in L requires that

$$ci + d = (ai + b)^2 = a^2 i^2 + 2abi + b^2.$$

From $i = 0$, we know $d = b^2$. Therefore it must also be the case that $ci = a^2 i^2 + 2abi$. But this implies that $i = (c - 2ab)/a^2$ for all $i \geq 0$ which is clearly false.