Problem 1.

Let $|x|_a$ be the number of occurrences of the symbol a in the string x.

- Define a context-free grammar for the language $L = \{w \in \{0,1\}^* : |w|_0 = |w|_1\}$.
- Given a formal proof that your grammar does indeed generate L.

Solution

$G = (\{S, Z, N\}, \{0,1\}, P, S)$ with productions P given by

$$S \to 0SNS \mid 1SZS$$
$$S \to \epsilon$$
$$Z \to 0$$
$$N \to 1$$

Proof. Let $w \notin L$, want to show that w is not generated by G.
By definition of L, w has an unequal number of 0's and 1's. Note that whenever G generates a 0 there must be a corresponding 1 (N) generated alongside of it, further whenever G generates a 1 there must be a corresponding 0 (Z) generated alongside of it as well. Via this paring strategy, strings generated by G are guaranteed to have equal number of 0's and 1's. Therefore, w cannot be generated by G, thus we only need to determine whether or not G can generate EVERY string of equal 0's and 1's, and not just a subset.
Let $w \in L$, want to show that w can be generated by G.
In order for w to be generated by G we need some pairing strategy which maps to the format of S for the strings in G. Note that there must be pairs of neighboring 0's and 1's in w, as w contains an equal number of both. If not, w is the empty string, and G holds no problem. We will work in an inside out strategy, that is, working from the bottom of the generated tree up. Choose one such pair of neighboring 0's and 1's, this pair can be generated by S using $0SNS \mid 1SZS$ with each $S \to \epsilon$. As these can be generated by S. In w, replace this pair with S, note that this working copy of w now has $|w|_a - 1$ pairs of a's and b's. Continue finding such pairs of 1's and 0's, but from now on, we can ignore any S (as we can use it instead of ϵ in the example above). We perform this process exactly $|w|_a$ times, until our string has been reduced to simply S, the starting non-terminal. Following this process in reverse order therefore generates w using G.

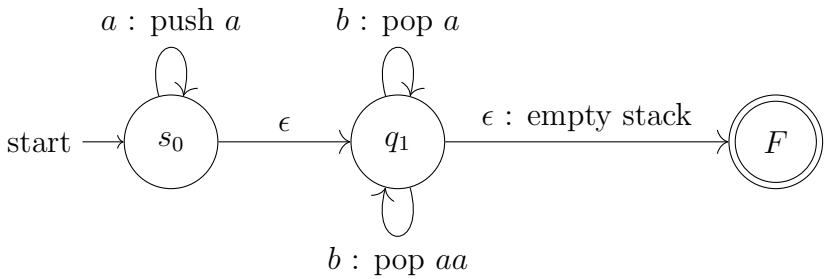
□

Problem 2.

Define a NPDA for the language $L = \{a^n b^m : m, n \in \mathbb{N}, m \leq n \leq 2m\}$.

Solution

The NPDA here is pretty simple, push every a we find on the stack, for each b there may either be 1 or 2 corresponding a's. Thus we have a split in the branching of the NDA, one which pops a single a off the stack on input of b, and the other which pops two a's off the stack on an input of b. Note that the empty string is also in this language, as it may make the both ϵ transitions as nothing is ever pushed on the stack.



Problem 3.

Define a NPDA for the language $L = \{uv \in \{0,1\}^* : |u| = |v| \wedge u \neq v^R\}$.

Solution

First note that $u \neq v^R \Rightarrow v \neq u^R$, we will take advantage of this by pushing u on the stack, then, by reading the stack, we are reading u^R. Once the stack is empty, we know that $|u| = |v|$, thus we need only verify that when popping u off the stack, we do so with at least one different character than what was found on the stack. Below, B represents the 'bad' state, where v has not yet proven itself unequal to u^R, as soon as we pop something different then we read, we enter G, the 'good' state, here we can read anything (where Z represents whatever happens to be at the top of the stack) until the stack is empty, and enter our final state. Note that ϵ is not in this language, as $\epsilon = \epsilon^R$, note that the empty string will never be able to make the distinguishing transition from the 'Bad' state to the 'Good' state, handling this case.