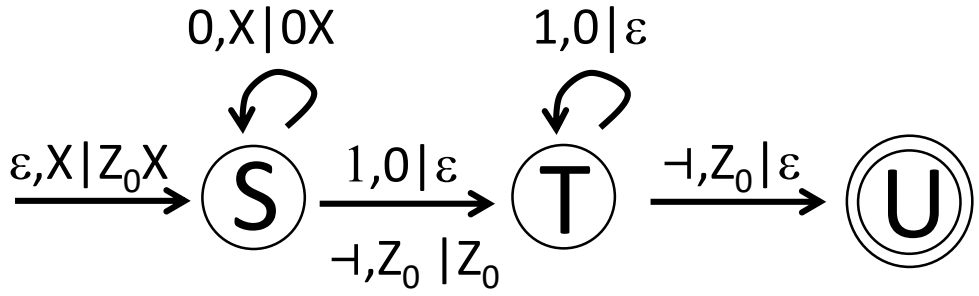
Deterministic PDAs

Before we leave the subject of context-free languages there is one more topic. A *deterministic* PDA is one in which there is exactly one choice of action for a given state, stack symbol and input symbol. By convention deterministic PDAs only accept by final state, but it is easy to show that is equivalent to accepting by empty stack. Since we no longer have non-deterministic rules we extend the input alphabet with symbol ⊣ and assume that is at the end of every string.

The languages accepted by deterministic PDAs are called deterministic context-free languages.

For example, the language $\{a^n b^n \mid n >= 0\}$ is deterministic context-free:



With regular languages nondeterminism doesn't add anything: any language accepted by an NFA is also accepted by a DFA. This is not the case with context-free languages: there are languages that are accepted by (standard, non-deterministic) PDAs that are not accepted by any deterministic PDA.

Here is an example. $\{ww^{rev} \mid w \epsilon (0+1)^*\}$, which is the language of even-length palindromes of 0s and 1s. This isn't a formal proof, but imagine a deterministic PDA that accepted this language. If it sees a string with prefix 00 it must save them on the stack, so it can accept if this prefix is followed by 00. But what does it do if the third character of the input is 0? If it pushes the 0 onto the stack it has no way to accept if the fourth and last character is 0 (so the full input is 0000). On the other hand, if it pops the stack it has no way to accept if the rest of the input is 11000 (so the full input is 00011000). There is no deterministic way to decide what to do with this third 0.

Deterministic Context-Free languages occupy a shadowy state between Regular languages (a DFA is a deterministic PDA that ignores its stack, so Regular languages are DCFL) and general Context-Free languages. There is no very satisfying description of Deterministic Context-Free languages, though there is a notion of a deterministic context-free grammar and a language is accepted by a deterministic PDA if and only if it has a deterministic grammar.

So why do we talk about them? It can be shown (and shouldn't be surprising) that a DCFL can be parsed in linear time. This is so important for compilers that all practical programming languages are based on DCFLs.

However, we can say the following:

Theorem: The complement of a DCFL is also a DCFL.
Proof: Just as with regular languages we want to accept a string if it takes us to something thing that is *not* a final state, but there are two problems:

A. There might be missing transitions, so the DPDA is unable to complete the computation.
B. Even in a DPDA the computation may be infinite because of loops of ε-transitions.

We can solve (A) by including a "dead" state, as we did with DFAs.

Loops are more complicated. Suppose the computation on input x goes into an infinite loop. At step i let γ_i be the stack contents. We can find a sequence of steps t0, t1, ... so that

$$|\gamma_{ti}| <= |\gamma_i| \text{ for all } i >= ti$$

(each ti is the point where $|\gamma_i|$ is minimized for i> ti-1)

So after step ti the stack never goes below the top symbol of γ_{ti}.

The sequence t0, t1, ... is infinite and there are only finitely many transitions, so some transition has to be applied infinitely often in this sequence of steps. Suppose this is the transition $(p,\varepsilon,A) \rightarrow (q,\beta)$.

To use this transition at step ti we must arrive in state p with A at the top of the stack. Because of the way the ti were chosen the computation never looks below symbol A on the stack. The computation must be independent of any input and it repeatedly gets back to state p with A on the stack. This means the transition $(p,\varepsilon,A) \rightarrow (q,\beta)$ can never be part of a string being accepted, so it can be replaced by

$(p,\varepsilon,A) \rightarrow (dead,A)$.

If we eliminate all such transitions the PDA will always proceed in a finite number of steps to either a final state on a non-final state. By taking the complement of the final states we get a DPDA that accepts the complement of the initial language.

So the family of deterministic context-free languages is closed under complements. Since this is not true for context-free languages in general, this means that there are context-free languages that are not deterministically context-free. In fact, it can be shown that $\{ww^{rev} | w \in (0+1)^*\}$ is not accepted by any DPDA so it is not deterministically context-free.