

Homework 6, CS301, McConnell, Spring '09

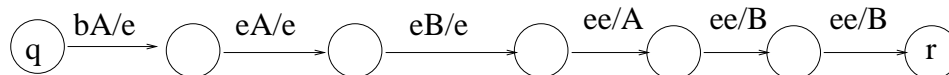
- Our book uses a model of a pushdown automaton that can pop multiple symbols and push multiple symbols with each transition. In the posting on converting a PDA to a context-free grammar, I used a model where you can pop at most one symbol and push at most one symbol with each transition. Let's call this a "one-at-a-time" PDA. Both of them accept if they end up with an empty stack after they've read the input. A skeptic could argue that maybe one-at-a-time PDA's always have context-free grammars, but PDA's of the book's might not. To discredit this argument, we need to have a proof up our sleeve that for every PDA of the book's type, there is a one-at-a-time PDA that accepts the same language. An acceptable proof would be an algorithm to convert a PDA of the book's type to a one-at-a-time PDA that accepts the same language.

One way to do this is to show how to convert each transition of one of the book's PDA's to a path of transitions of the one-at-a-time variety. If you figure out a way to do this, draw the path that your algorithm would convert the following transition to:



A transition that reads a b
from the input, pops AAB,
and pushes BBA

Solution:

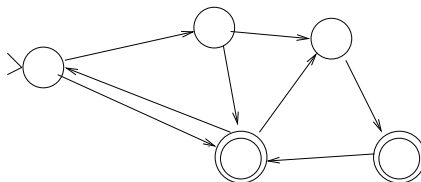


- Prove that for every one-at-a-time PDA, there is a PDA of the book's variety that accepts the same language. (There is a very short proof of this.)

Solution: *The one-at-a-time variety is a special case of the book's variety.*

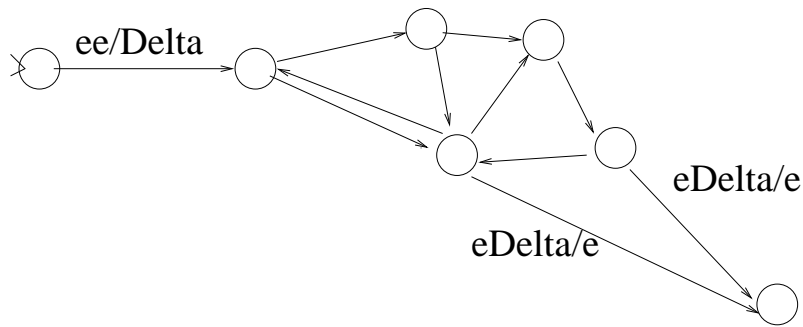
- The initial definition of the PDA on pages 81-82 requires the PDA to be in an accept state with an empty stack when the string has been read in order to accept the string. Theorem 3.3.1 of the book shows that for every PDA of this type, there is a PDA that only requires the stack to be empty.

Illustrate the transformation given by the proof on the following example:



I would write what's read, pushed, and popped on the transitions, but it is irrelevant to the transformation, which preserves those transitions and whatever action they take.

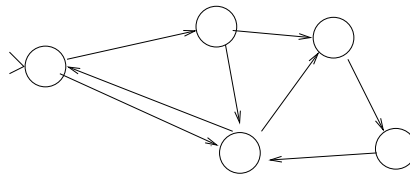
Solution:



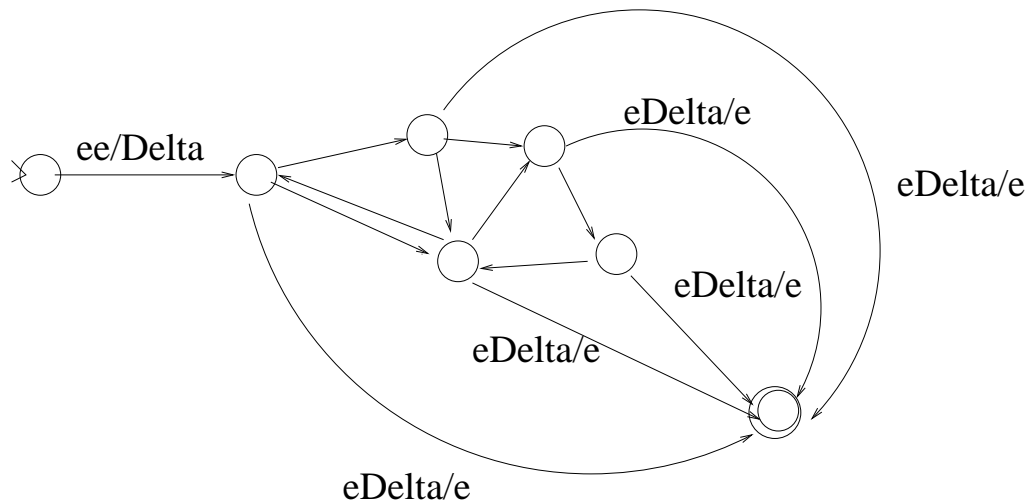
4. Prove the converse of Theorem 3.3.1. That is, prove that for every PDA that requires only that the stack be empty, there is one that requires both that the stack be empty and that you be in an accept state. *Hint: This is a very short proof.*

Solution: Take one that only requires that the stack be empty, and make all the states accept states; a word is accepted by the new one on empty stack and accept state if and only if it's accepted on the old one on empty stack.

5. Think of a proof that for every PDA that accepts if the stack is empty, there is a PDA that accepts if you're in an accept state, whether or not the stack is empty. Illustrate your transformation on the following example:

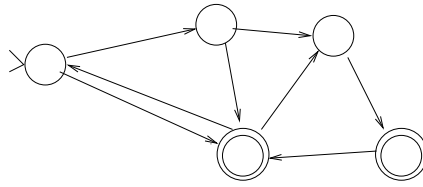


Solution:



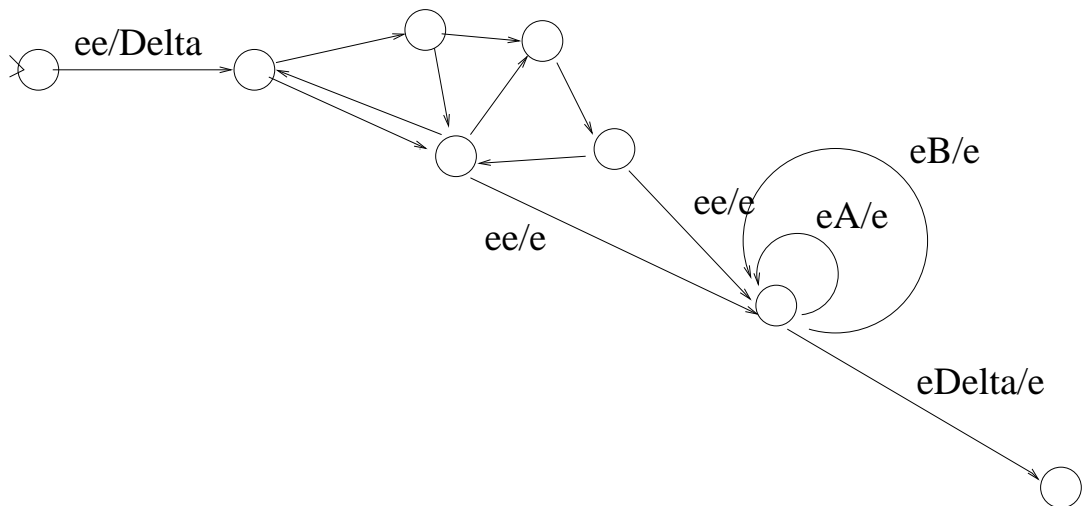
Hint: the proof is similar to that of Theorem 3.3.1.

6. Think of a proof of the converse of this. Illustrate the transformation on the following example:



(This looks like the one in a previous picture, but the previous picture depicted one that required ending in an accept state *and* having an empty stack. This one is supposed to represent one where you only have to end in an accept state.)

Solution:

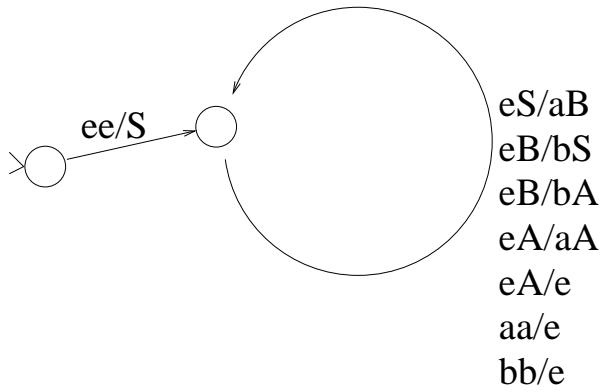


7. Theorem 3.4.1 gives an algorithm that proves that for every context-free language, there is a PDA that accepts the language.

Illustrate the transformation on the language generated by the following context-free grammar, by drawing the PDA:

$$\begin{aligned}
 S &\rightarrow aB|e \\
 B &\rightarrow bS|bA \\
 A &\rightarrow aA|e.
 \end{aligned}$$

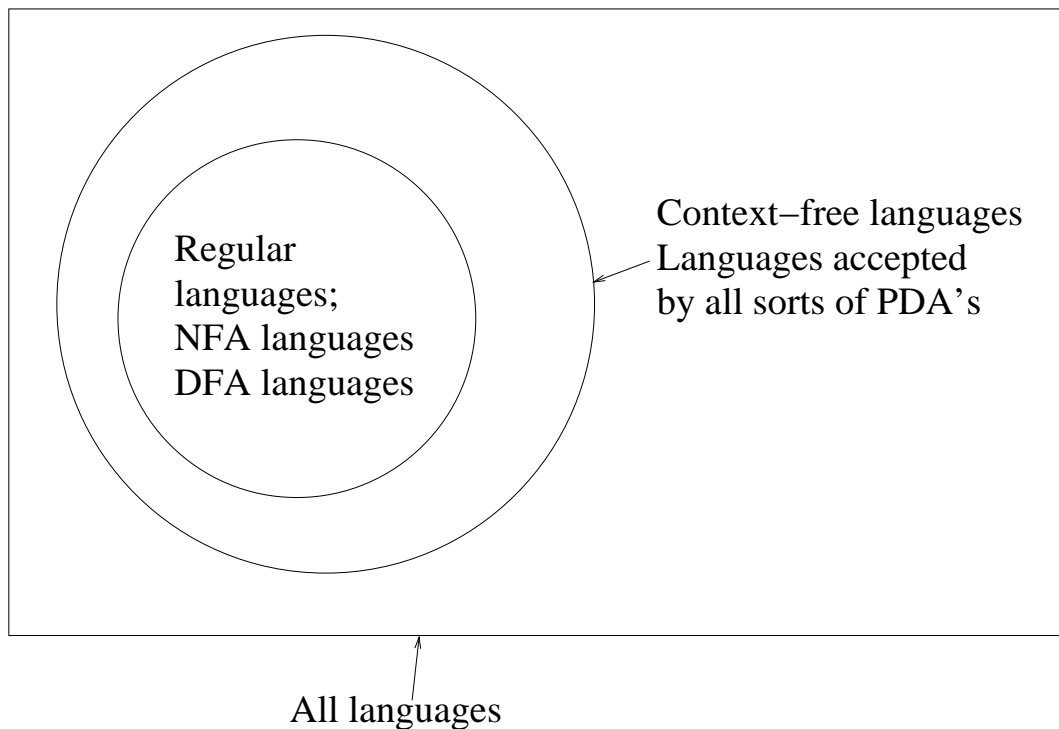
Solution:



When it says aB is pushed, this means that B is pushed first, then a .

8. We've gotten a picture of the landscape of these these regular and context-free languages, NFA's, DFA's, and PDA's,
- We showed that the regular languages are exactly the same as those that are accepted by NFA's, which are exactly the same as those that are accepted by DFA's. (Review how we did this.)
 - We showed that some languages are not regular by showing that the set of all languages on $\{a, b\}$ is uncountably infinite, and the set of all regular expressions are countably infinite. (Review this.)
 - This was a non-constructive proof, because it showed that some non-regular languages exist, but didn't show us how to find any. We showed a way to find some of them (the pumping lemma for regular languages.)
 - Once we did this, we saw that some of the examples of non-regular languages it gave us are context-free, by giving grammars for them. This opened the question of whether there are also non-context-free languages that are regular. We showed that this is not the case. (Review how we did this. We saw several proofs.) This implies that the regular languages are a proper subset of the context-free languages.
 - We showed that for every context-free language, there is a PDA of the book's type that accepts that language. The construction is the one you illustrate in the last problem.
 - You showed above that many variants of PDA's accept the same set of languages, and these include the book's. This means that everything the book says about it's PDA's applies to each of the varieties you dealt with above.
 - The posting on converting a PDA to a context-free grammar showed that for one of these variants, there is a context-free grammar for that PDA. We concluded that the languages accepted by all these PDA's are precisely the context-free languages.
 - At this point, we had not established that there are non-context-free languages. We used the pumping lemma for context-free languages to show this.

This gives us the following picture of the set of languages:



Suppose we didn't know about the pumping lemma for context-free languages. Give a non-constructive proof that there must exist languages that are not context-free. *Hint: review the non-constructive proof that there are non-regular languages.*

Solution: *There are a countably infinite set of grammar's on terminal alphabet $\{a, b\}$, since each can be encoded with a string of symbols on an alphabet that includes symbols such as \rightarrow and $|$ and nonterminals, which, as we have seen in the handout, can each be encoded on a string of symbols that include $[,]$, etc., that can be typed on a keyboard. Each grammar can be expressed as a string of characters on the alphabet of the keyboard. The set of all grammars is a single language on that alphabet. Like all languages, its size is countably infinite. There must be a countably infinite set of context-free languages. As we have seen, the set of all languages is uncountably infinite. There must be non-context-free languages.*