Format for describing PDAs

A PDA is defined by (i) its initial stack symbol, (ii) its initial state, (iii) its set of accepting states, and (iv) its transition rules. Hence, in order to describe a PDA we write in the first line the information corresponding to (i), (ii) and (iii), and in the successive lines we write the transition rules. For instance, with a first line like:

```
Z q0 q0 q3
```

we specify that (i) the initial contents of the stack is the symbol `Z`, that (ii) the initial state of the automaton is `q0`, and (iii) that the accepting states are `q0` and `q3`. After that line, we need to specify each of the transitions rules. To this end, we may use two different syntaxes, which we describe with examples:

- `Z q1 a -> ZA q1`
  This describes a transition from state `q1` when the stack top is the symbol `Z` and the input is the symbol `a`. Executing this transitions would read the next input symbol (which in this case is `a`), pop the top of the stack (which in this case is `Z`), push onto the stack `ZA` (implying that we recover the read `Z` by pushing it again, and then push an `A` onto it) and change the state from `q1` to `q1` again.

  The symbol read (`a` in the example) as well as the symbols pushed onto the stack (`ZA` in the example) are optional.

- `q1 -> Za|ZA -> q1`
  The interpretation of this transition is equivalent to the previous one. The benefit of this alternative syntax is that it allows to write several transitions together if they share the same origin state and the same destiny state. For instance, if the automaton also had a transition `q1 -> Aa|AA -> q1`, we could write both of them together like `q1 -> Za|ZA, Aa|AA -> q1`.

  As before, recall that the symbol read as well as the contents pushed onto the stack are optional for each transition.

In order to conclude the explanation, we introduce a complete example automaton. The following PDA recognizes the language $\{\texttt{a}^n\ \texttt{b}^n \mid n \geq 0\}$:

```
Z q0 q0 q3
Z q0    -> Z  q1
Z q1 a -> ZA q1
A q1 a -> AA q1
A q1 b ->    q2
A q2 b ->    q2
Z q2    -> Z  q3
```

Note that the fourth and fifth rules do not push any symbol onto the stack, and that the first and last rules do not read any input symbol. This example PDA is *deterministic*, i.e., given a stack symbol, an input symbol and a state, there is at most one transition rule that can be executed. Moreover, it is also *uniquely accepting* since there is only one accepting execution of the automaton for each word in the recognized language. As a final remark, using the second syntax for the rules, the automaton can be equivalently written as:

```
Z q0 q0 q3
q0 -> Z |Z          -> q1
q1 -> Za|ZA, Aa|AA -> q1
q1 -> Ab|           -> q2
q2 -> Ab|           -> q2
q2 -> Z |Z          -> q3
```

where it is easy to see the correspondence with the graph traditionally used to represent PDAs: