Problem

Let a k-PDA be a pushdown automaton that has k stacks. Thus a 0-PDA is an NFA and a 1-PDA is a conventional PDA. You already know that 1-PDAs are more powerful (recognize a larger class of languages) than 0-PDAs.

a. Show that 2-PDAs are more powerful than 1-PDAs.

b. Show that 3-PDAs are not more powerful than 2-PDAs.

(Hint: Simulate a Turing machine tape with two stacks.)

Step-by-step solution

Step 1 of 3

Push down automata:

Push down automata (PDA) is like nondeterministic finite automata but have an extra component called stack. Stack provides additional memory beyond the finite amount available in the control.

• *k*-PDA is a push down automaton that has $k-$ stacks

• $0-\text{PDA}$ is a NFA

• $1-\text{PDA}$ is a conventional PDA.

It is known that $1-\text{PDAs}$ are more powerful than $0-\text{PDAs}$.

It means $1-\text{PDAs}$ are recognized by a larger class of languages than $0-\text{PDAs}$.

Comment

Step 2 of 3

(a)

Now it is required to show that $2-\text{PDAs}$ are more powerful than $1-\text{PDAs}$.

Prove this by taking an example.

As it is known that “A Language is context free if and only if some pushdown automaton recognizes it.”

Let $B=\left\{a^n b^n c^n \mid n \geq 0\right\}$ be the language.

P is the pumping length of the B given by the pumping lemma.

Now prove that B is not a CFL (Context-Free Language).

To show that B is not a CFL, it’s enough to show that a string $s=1^p 0^p 1^p 0^p$ cannot be pumped.

Consider *s* is of the form *s=uvxyz*.

• If both *v* and *y* contain at most one type of alphabet symbol, the string will be of the form uv^2xy^2z runs of $0\text{'s and }1\text{'s}$ of unequal length. Hence the string *s* cannot be a member of B.

• If either *v* or *y* contains more than one type of alphabet symbol, the string will be of the form uv^2xy^2z which does not contain the symbols in correct order. Hence the string *s* cannot be a member of B.

Since the string *s* cannot be pumped without violating the pumping lemma condition, *B* is not a CFL (context-free language). Thus $1-\text{PDA}$ does not recognize *B*.

But the following $2-\text{PDA}$ recognizes *B*.

• Push all $a's$ that appear in front of the input tape to stack 1.

• Then push all $b's$ that follow the $a's$ in the input stream into stack2. If it sees any $a's$ at this stage, ***reject.***

• When it sees the first *c*, pop stack 1 and stack2 at the same time. After this, reject the input if it sees any $a's$ (or) $b's$ in the input stream.

• Pop stack1 and stack2 for each input character *c* it reads. If the input ends when both stacks are empty, ***accept***. Otherwise ***reject.***

Therefore, B is recognized by $2-\text{PDA}$.

Thus $2\text{PDA}s$ are more powerful then $1-\text{PDA}s$.

Comments (2)

Step 3 of 3

(b)

To show that 3-PDAs are not more powerful than 2PDAs,

It is known that 3-PDA is not powerful than Turing machine, so it is required to show that the $2-\text{PDA}$ can simulate a Turing machine, then obviously 3-PDAs are not powerful than 2-PDAs.

Simulation of TM by 2PDA:

Now split the tape of TM (Turing machine) into two stacks.

• Stack1 stores the characters on the left of the head, with the bottom of stack storing the left most character of tape in the TM.

• Stack 2 stores the characters on the right of the head, with the bottom of the stack storing the right most character on the tape in the TM.

Show the Transition example for left and right.

Left Transition:

• If $\delta(q_i, w_i) = (q_j, w_j, L)$, then w_i on top of second stack.

• If $ is on top of stack then reject. Move left edge of tape.

o Otherwise pop w_i from top of second stack. Push w_j in second stack.

• Pop the symbol of first stack and push it to second stack. Switching the state of machine q_j.

• If q_j is accept state, then accepts *w*. If q_j is reject state, then rejects *w.*

Right Transition:

• If $\delta(q_i, w_i) = (q_j, w_j, R)$, then w_i on top of second stack.

• Pop w_i from second stack and push w_j in second stack

• If $ is on top of second stack, push null on to second stack. Either case, switching the state of machine q_j.

• If q_j is accept state, then accepts *w*. If q_j is reject state, then rejects *w.*

In this way $2-\text{PDA}$ simulates the Turing Machine.

Furthermore,

Now simulate 4-tape non-deterministic Turing machine with 3-PDA.

• First tape as input tape, second, third, and fourth tapes respectively first, second and third stack.

• If stack changed by push without pop then N moves the tape head to right without change the current symbol. Now write the symbol to the tape which is pushed on stack.

• If stack is changed by pop without push, then N replaces tape head with blank symbol and moves left.

• If stack is changed by push and pop, then N changes the tape by written symbol pushed in current cell.

• If stack is not changed, then N writes same symbol on tape head and then back to tape.

• Symbol is used from the tape then N moves first tape to right.

• If symbol is not used then N copies the element at first tape head.

• These actions can parallel in all tapes at same time since they take only one turn.

• 2nd point mentioned Push without pop, in this case all tape heads kept same except stacks affected by push without pop. Moves right to get the blank symbol at tape.

• N is nondeterministic to guess which transition to take same way PDA does.

So, it proves that Turing Machine are powerful as 3-PDAs, so it can be concluded that 3-PDAs are no more powerful than 2-PDAs. Both the PDAs are equally powerful.

Comments (1)