

Other Automata

We can create PDAs with multiple stacks. At each step we look at the current state, the current input symbol, and the top of each stack. From all of this information we decide what state to transition to and what to write on each stack.

Theorem: A TM can simulate a PDA with k stacks.

Proof: Use a TM with $k+1$ tapes: one for the input, the other tapes each representing one stack.

Theorem: A PDA with 2 stacks can simulate a TM.

Proof: Suppose the current configuration is $10BXq00Y$

We will store this as

X	
B	0
0	0
1	Y
Z_0	Z_0
stack 1	stack 2

The current tape symbol is always at the top of stack 2; the symbol to its left is at the top of stack 1. We move Left on the tape by popping stack 1 onto stack 2; we move Right by popping stack 2 onto stack 1. The PDA controller can simulate the actions of the TM controller: read the tape symbol (top of stack 2), write a new tape symbol (pop stack 2, push new symbol) and move left or right.

Counter Machines

A k-counter machine has a finite state controller and k counters in place of a stack or tape. Each counter is a non-negative integer. The only operations with a counter are

- Add 1 to the counter
- Subtract 1 from the counter
- Test the counter for being 0

Note that we can represent a counter with a stack. To increment the counter push something onto the stack; to decrement the counter pop the stack; to test for 0 ask if the stack is empty.

Theorem: Any language accepted by a 1-counter machine is context-free.

Theorem: Any language accepted by a k -counter machine is recursively enumerable (accepted by a TM)

We will now show that we can simulate a TM with a 3-counter machine. We know we can simulate a TM on a PDA with two stacks so we'll show we can simulate the latter with 3 counters.

Suppose the stack alphabet has $(r-1)$ symbols. Identify these symbols with the digits $1\dots(r-1)$. (We deliberately don't use 0.) Then represent the stack

$$X_1$$
$$X_2$$
$$\dots$$
$$X_n$$
$$Z_0$$

by $X_1 + X_2r + X_3r^2 + \dots + X_nr^{n-1}$; i.e., treat the stack as a number in base r .

For example, suppose we have 9 stack symbols A,B,C,D,E,F,G,H, I, which we identify with 1,2,...9. Then the stack

C

B

B

C

A

Z_0

is represented by 13223.

To push symbol X onto the stack represented by a counter, multiply the counter by r and add X .

To pop the stack, divide by r and ignore any remainder.

To get the value at the top of the stack, take the counter mod r .

To simulate 2 stacks with 3 counters, we will use 2 counters to represent the contents of the two stacks and keep the other counter free.

1. We can have states that add a fixed number to a counter (by doing that many increments).
2. To multiply a counter A by r , first empty the free counter. Add r to the free counter and decrement A . Again add r to the free counter and decrement A . Continue until A is 0. The free counter now has $A*r$. Copy this to counter A .
3. To divide counter A by r , first empty the free counter. Decrement A r times then increment the free counter. Repeat until A can't be decremented r times. The value in the free counter is A/r .

4. We need to be able to compute $A \% r$ without changing A . Make a cycle of r nodes $m_0 m_1 \dots m_{r-1} m_0$. Empty the free counter. Walk around the cycle of nodes at each step decrementing A and incrementing the free counter. When A is empty the free counter has the original value of A and the index i of m_i is the value of $A \% r$. So copy the free counter to A and then increment the free counter i times.

These operations let us simulate the top, push and pop operations of two stacks with 3 counters.

Theorem: We can simulate a 3-counter machine with just 2 counters.

Proof: Suppose the 3 counters have values i, j, k . Represent all three with one counter $2^i 3^j 5^k$.

- a) To increment a counter multiply by 2, 3, or 5.
- b) To decrement a counter, divide by 2, 3, or 5.
- c) To determine if a counter is 0, look at the remainder when dividing by 2, 3, or 5. If the remainder is 0, the counter is not 0.

Moral: a 2-counter machine is equivalent to a TM.