Today's lectures: Fun With Turing Machines

Reading:

November 2: Fun With Turing Machines
Read: *Kozen* lecture 29 (or *Sipser* 3.1 again).

November 4: Review and Examples.

November 7: Modified Turing Machines
Read: *Kozen* lecture 30 (or *Sipser* 3.2).

November 9: Second Midterm: in class.

November 14: Diagonalization and the Halting Problem
Read: *Kozen* lecture 31 (or *Sipser* 4.2).

November 16: Decidability
Read: *Kozen* lecture 32 (or *Sipser* 4.1).

November 18: Review and Examples.

November 21: Reductions
Read: *Kozen* lecture 33 (or *Sipser* Chapter 5).

November 23: Gödel's Theorem
Read: *Kozen* lecture 38 (or *Sipser* 6.2).

November 25: Review and Examples.

November 28: Everything Else About Turing Machines.

November 30: Theorem Proving.

December 2: Something Fun.

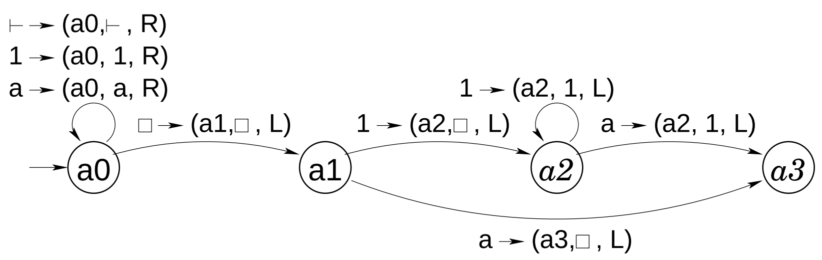

Figure 1: A Turing machine that adds two numbers together

I. Addition

A. The tape format

1. As usual, $\{\vdash, \Box\} \in \Sigma$, where

a. $\vdash$ is the left end-marker.

b. $\Box$ is a blank space

2. We will represent non-negative integers in unary. Let $1 \in \Sigma$. The string 1^n represents the integer n.
We use unary because it makes the design of the other operations described below simpler. It is possible to use binary (or any other base, for that matter). Real machines use binary because unary requires exponentially more space for storage. Because in this class we mainly worry about what can be computed and we rarely worry about effiency, we'll use unary.

3. We will also include the symbol $ain\Sigma$.

a. We'll use a as a separator between integers symbols.

b. For example, the tape:

$$\vdash 11a111aa1\Box^\omega$$

encodes the sequence of integers: $2, 3, 0, 1$.

B. Our machine will replace the last two numbers on the tape with their sum.

1. The basic plan:

a. Move to the right until a $\Box$ is encountered.

b. Shift symbols to the left until the rightmost a has been overwritten by the symbol to its right.

c. Example: $\vdash 1aa11a111\Box^\omega$ will become $\vdash 1aa11111\Box^\omega$. This corresponds to the famous result from number theory: $2 + 3 = 5$.

2. The machine

a. See figure 1. Omitted transitions should never occur and can be filled in arbitrarily.

b. An equivalent program: I'll assume a function moveRight(state, symbol) that writes symbol, moves the read/write head one position to the right, and transitions to state state. If state is omitted, the current state is assumed. Likewise, if symbol is omitted, the current symbol is written back onto the tape. The function moveLeft works in the same way, except that the read/write head moves to the left.

```
a0: while(currentSymbol() != ' ') moveRight();
    moveLeft();
a1: if(currentSymbol() == '1') moveLeft(a2, □);
    else if(currentSymbol() == 'a') moveLeft(a3, □);
a2: while(currentSymbol() == '1') moveLeft();
    if(currentSymbol() != 'a') fatalError();
    else moveLeft(a3, 1);
a3:
```

c. Solving that famous number theory puzzle – see figure **??**.

II. Multiplication: our machine will replace the last two numbers on the tape with their product.

A. Append the number zero to the tape.

Step	State	Tape	Step	State	Tape	Step	State	Tape
0	a_0	$\underline{\vdash} 1aa11a111\Box^\omega$	5	a_0	$\vdash 1aa1\underline{1}a111\Box^\omega$	10	a_1	$\vdash 1aa11a11\underline{1}\Box^\omega$
1	a_0	$\vdash \underline{1}aa11a111\Box^\omega$	6	a_0	$\vdash 1aa11\underline{a}111\Box^\omega$	11	a_2	$\vdash 1aa11a1\underline{1}\Box\Box^\omega$
2	a_0	$\vdash 1\underline{a}a11a111\Box^\omega$	7	a_0	$\vdash 1aa11a\underline{1}11\Box^\omega$	12	a_2	$\vdash 1aa11a\underline{1}1\Box^\omega$
3	a_0	$\vdash 1a\underline{a}11a111\Box^\omega$	8	a_0	$\vdash 1aa11a1\underline{1}1\Box^\omega$	13	a_2	$\vdash 1aa11\underline{a}11\Box^\omega$
4	a_0	$\vdash 1aa\underline{1}1a111\Box^\omega$	9	a_0	$\vdash 1aa11a11\underline{1}\Box^\omega$	14	a_3	$\vdash 1aa1\underline{1}111\Box^\omega$

Figure 2: Configurations for computing $2 + 3$

B. Let z denote the number at the right end of the tape.
Let y denote the number one to the left of the right end of the tape.
Let x denote the number two to the left of the right end of the tape.
In other words, our tape looks like: $\vdash \ldots a1^x a1^y a\Box^\omega$ (because z is initially 0).

C. The multiplication loop:

```
for each 1 in $x$
  decrement $x$;
  append a copy of $1^y$ to the tape;
  add the last two numbers on the tape together;
end;
```

D. Finally, we delete x and y.

III. Factorial

A. First, we'll show how to implement subroutines.

1. The tape format

a. We'll add a new symbol b to the tape alphabet.

b. The tape will now look like: $\vdash 1^i a1^j a1^k \ldots b1^m a1^n \ldots \Box^\omega$

c. The numbers to the left of the b are return addresses (i.e. numbers encoding machine states).

d. The numbers to the right of the b just as before.

2. Calling a subroutine

a. Append the state from which the machine should continue execution *after* the subroutine to the list of numbers to the left of the b.

b. Move to the state that's the entry point for the subroutine.

3. Returning from a subroutine.

a. Move the read/write head to the last number before the b symbol.

b. Read that number and move to the corresponding state.

c. The first sequence of operations in the return state should delete itself from the list of return addresses.

B. Now, we'll describe a machine that replaces the rightmost number on the tape with its factorial.

IV. Terminology

A. A Turing machine (TM) is total if for every string in Σ^*, the TM eventually accepts or rejects.

B. A set is *recursive* if there is a total TM that accepts it.

C. A set is *recursively enumerable* if there is a TM that successively outputs each member of the set.

D. A property is *decidable* if the set of strings having the property is recursive.

E. A property is *semidecidable* if there is a TM that accepts every string in the set of strings having the propery and either rejects or fails to terminate for all other strings. If a property is semi-decidable, then the set of strings with the property is recursively enumerable.