Turing Machines

Sipser pages 137-147

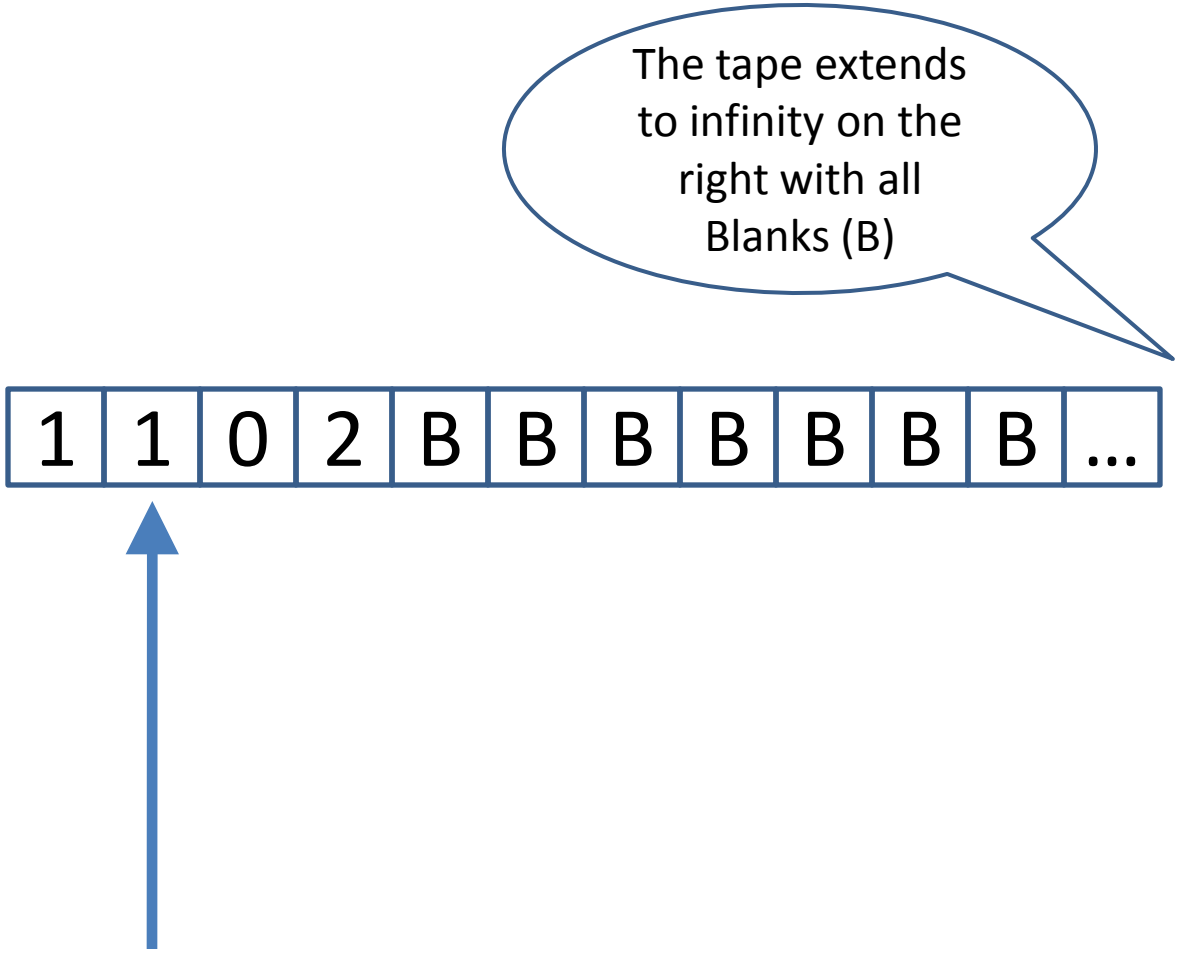
Intro to Turing Machines

- A *Turing Machine* (TM) has finite-state control (like PDA), and an infinite read-write *tape*. The tape serves as both input and unbounded storage device.

- The tape is divided into *cells*, and each cell holds one symbol from the *tape alphabet*.

- There is a special *blank* symbol B. At any instant, all but finitely many cells hold B.

- *Tape head* sees only one cell at any instant. The contents of this cell and the current state determine the next move of the TM.



state	0	1	2
1	L,2,3		
2			
3			
4			

Moves

- A *move* consists of:
  - replacing the contents of the scanned cell
  - repositioning of the tape head to the nearest cell on the left, or on the right
  - changing the state

- The *input alphabet* is a subset of the tape alphabet. Initially, the tape holds a string of input symbols (the *input*), starting on the left of the tape, with in infinite sequence of blanks to the right (after the input). The initial position of the head is the leftmost symbol.

Formal Definition

- A TM is a 7-tuple $M=(Q,\Sigma,\Gamma,\delta,q_0,q_{accept},q_{reject})$, where

1. Q is a finite set of states
2. Σ is the input alphabet (does not contain B the blank symbol)
3. Γ is the tape alphabet, where
   1. $\Sigma \subseteq \Gamma$ (the input alphabet is a subset of the tape alphabet)
   2. B $\in \Gamma$ (Blank is in the tape alphabet)
4. $q_0 \in Q$ is the start state
5. $Q_{accept} \in Q$ is the accepting state
6. $Q_{reject} \in Q$ is the rejecting state
7. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$ is a partial function.

   The value of δ (q,X) is either undefined, or is a triple consisting of the new state, the replacement symbol, and direction (left/right) for the head motion.

Example

- Here is a TM that checks its third symbol is 0, accepts if so, and runs forever, if not (note it never rejects).

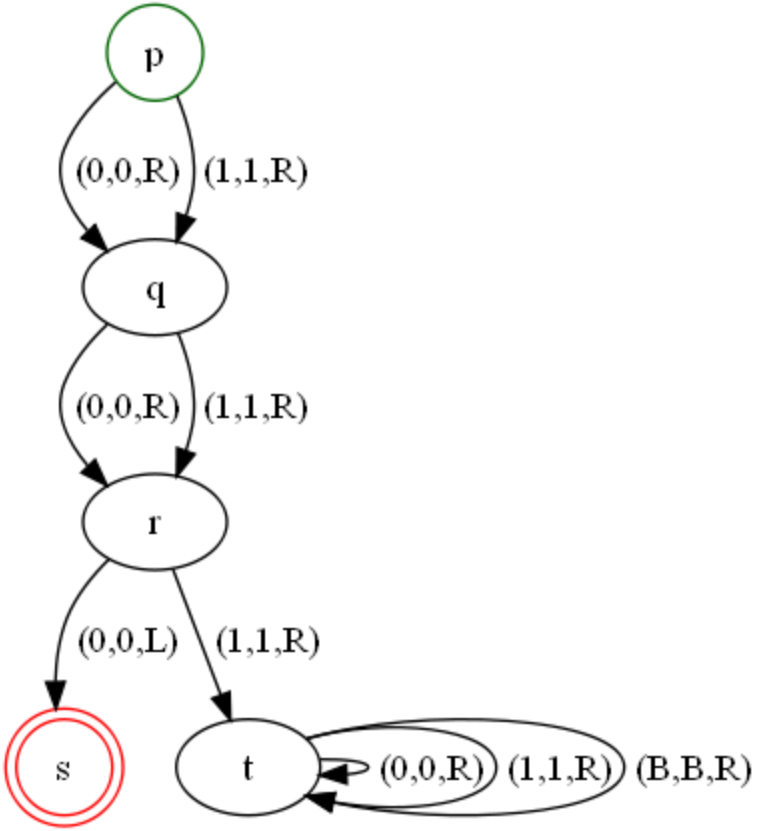
- M=({p,q,r,s,t,d},{0,1,},{0,1,B},p,s,d)

- δ(p,X) = (q,X,R) for X=0,1
- δ(q,X) = (r,X,R) for X=0,1
- δ(r,0) = (s,0,L)
- δ(r,1) = (t,1,R)
- δ(t,X) = (t,X,R) for X=0,1,B

Transisition Diagrams for TM

- Pictures of TM can be drawn like those for PDA's. Here's the TM of the example below.

$\delta(p,X) = (q,X,R)$ for X=0,1
$\delta(q,X) = (r,X,R)$ for X=0,1
$\delta(r,0) = (s,0,L)$
$\delta(r,1) = (t,1,R)$
$\delta(t,X) = (t,X,R)$ for X=0,1,B



Implicit Assumptions

- Input is placed on tape in contiguous block of cells (all the way to the left)
- All other cells to the right are blank: ‘B’
- Tape head positioned at Left of input block
- There is one start state

- The text uses a single accepting and rejecting state, an alternative is to have many such states. These are equivalent, why?

Example 2: $\{ a^n b^m \mid n,m \geq 0\}$

states = 0,1,H

tape alphabet = a,b,^

input alphabet = a,b

start = 0

blank = '^'

accepting = H

rejecting = R

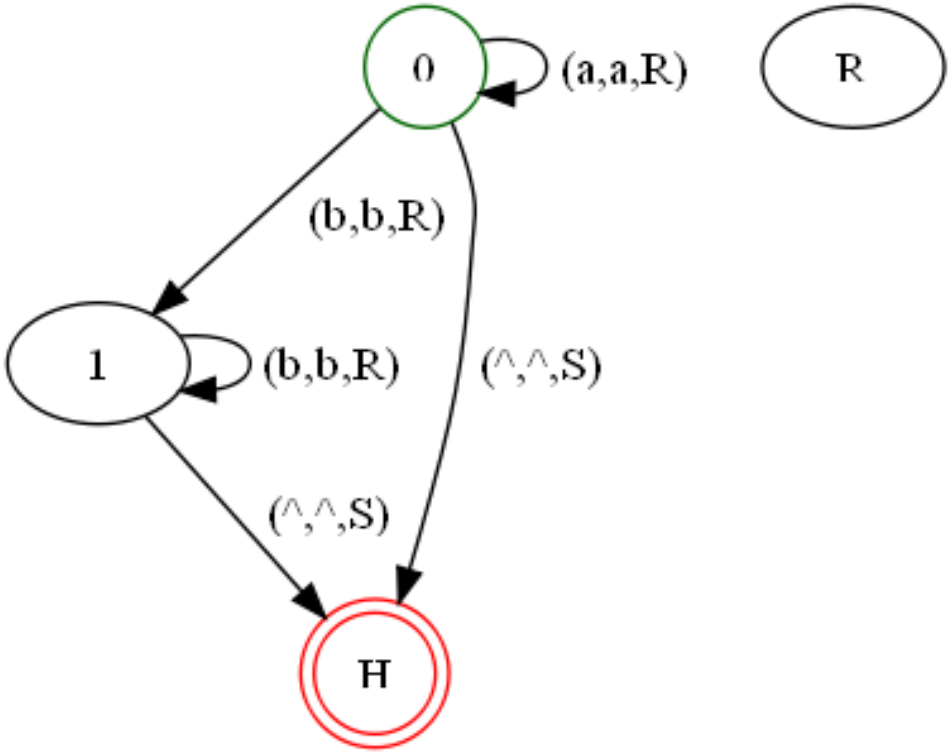
delta = (0,^,^,S,H)

(0,a,a,R,0)

(0,b,b,R,1)

(1,b,b,R,1)

(1,^,^,S,H)

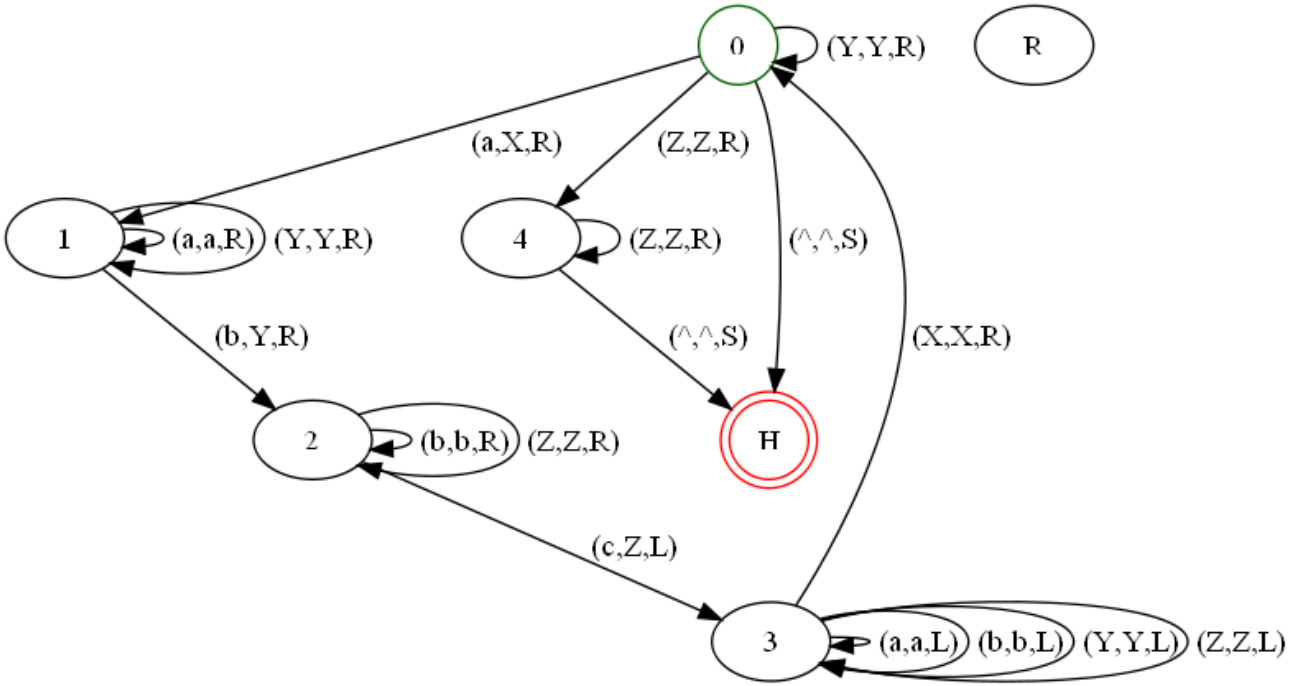


Example 3: $\{ a^n b^n c^n \mid n \geq 0 \}$

delta =

(0,a,X,R,1) Replace a by X and scan right
(0,Y,Y,R,0) Scan right over Y
(0,Z,Z,R,4) Scan right over Z, but make final check
(0,^,^,S,H) Nothing left, so its success
(1,a,a,R,1) Scan right looking for b, skip over a
(1,b,Y,R,2) Replace b by y, and scan right
(1,Y,Y,R,1) scan right over Y
(2,c,Z,L,3) Scan right looking for c, replacxe it by Z
(2,b,b,R,2) scan right skipping over b
(2,Z,Z,R,2) scan right skipping over Z
(3,a,a,L,3) scan left looking for X, skipping over a
(3,b,b,L,3) scan left looking for X, skipping over b
(3,X,X,R,0) Found an X, move right one cell
(3,Y,Y,L,3) scan left over Y
(3,Z,Z,L,3) scan left over Z
(4,Z,Z,R,4) Scan right looking for ^, skip over Z
(4,^,^,S,H) Found what we're looking for, success!

tape alphabet = a,b,c,^,X,Y,Z
input alphabet = a,b,c
start = 0
blank = '^ '
final = H



aabbcc
Xabbcc
XaYbcc
XaYbZc
XXYbZc
XXYYZc
XXYYZZ

Details

- There are 3 level of details we might use to describe TM's

1. Give the complete formal description
   - the states, alphabet, transition, etc
2. Implementation description
   - Partition the states into stages, Use English to describe how each stage move the head and stores data on the tape
   - Each stage performs one function.
   - Describe how we move between stages
   - No details of states or transition function
3. High level description
   - Use English to describe an algorithm
   - Don't mention how to describe tape or moves

Example Implementation level

- $\{ 0^{2\hat{}n} \mid n \geq 0\}$
- Stages
  1. Sweep left to right across the tape. Crossing off every other 0
  2. If in stage 1, the tape conatined a single 0, accept
  3. If in stage 1, the tape contained more than a single 0, and the number of 0's was odd, reject
  4. Return the head to the left-hand end of the tape
  5. Goto stage 1.

What is important

- We seek to convince the student that Turing machines are a powerful tool to describe algorithms
- The full set of details is often too complex to describe completely, because the details do not add to our understanding.
- But, we could use our high level descriptions to complete the formal description if we desired.

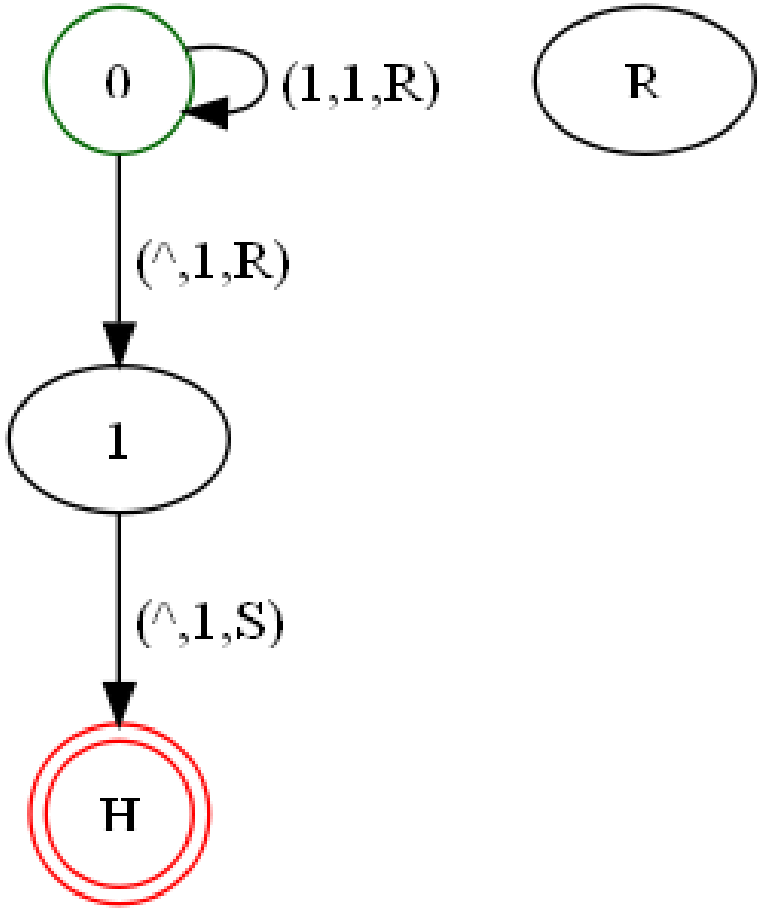
Turing machines with output

- A Turing machine can compute an output by leaving an answer on the tape when it halts.

- We must specify the form of the output when the machine halts.

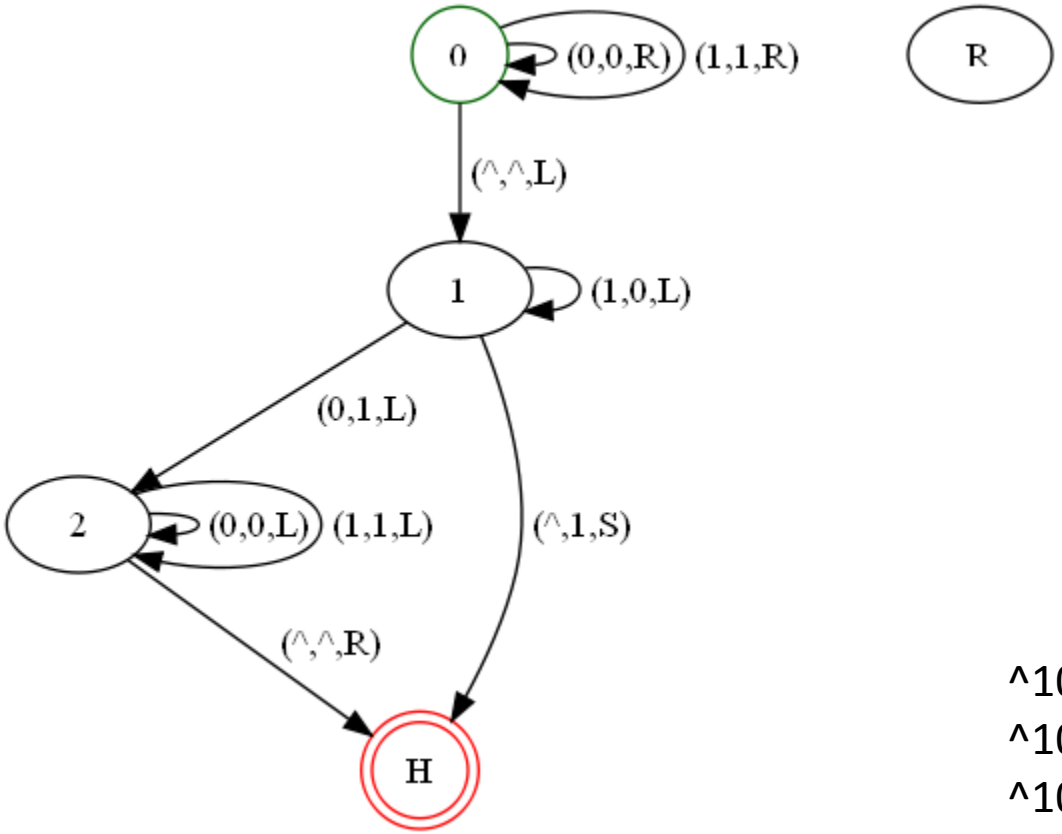
Adding two to a number in unary

```
TM    Q       {0, 1, H, R}
      Sigma   {1}
      Gamma   {1, ^}
      Delta   0 1 ->  (1, R, 0)
              0 ^ ->  (1, R, 1)
              1 ^ ->  (1, S, H)
      q0      0
      Accept  H
      Reject  R
      Blank   ^
```



Adding 1 to a Binary Number

TM Q {0, 1, 2, H, R}
Sigma {1, 0, ^}
Gamma {1, 0, ^}
Delta 0 0 -> (0, R, 0)
0 1 -> (1, R, 0)
0 ^ -> (^, L, 1)
1 0 -> (1, L, 2)
1 1 -> (0, L, 1)
1 ^ -> (1, S, H)
2 0 -> (0, L, 2)
2 1 -> (1, L, 2)
2 ^ -> (^, R, H)
q0 0
Accept H
Reject R
Blank ^

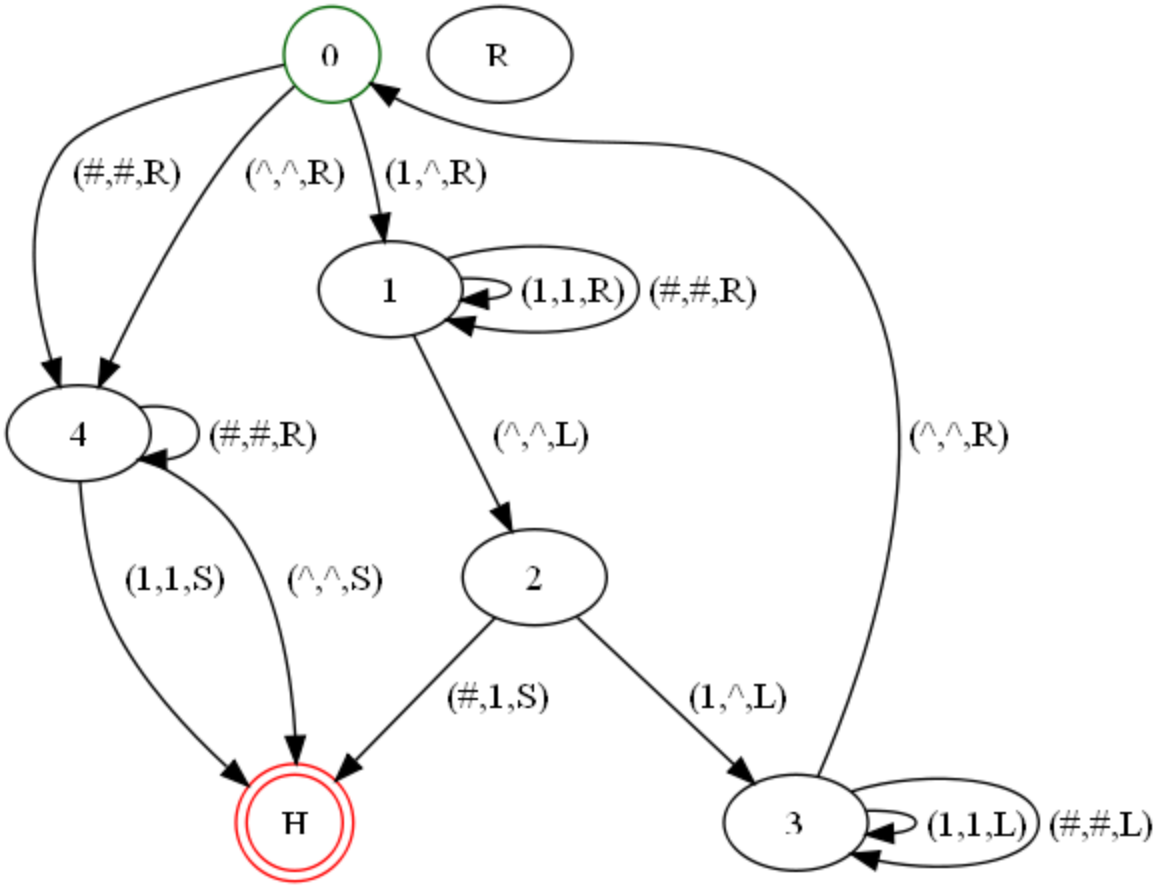


^1011^
^1010^
^1000^
^1100^

An equality Test

states = 0,1,2,3,4,H
tape alphabet = 1,0,#,^
input alphabet = 1,0,#
start = 0
blank = '^'
final = H

delta =
(0,1,^,R,1)
(0,^,^,R,4)
(0,#,#,R,4)
(1,1,1,R,1)
(1,^,^,L,2)
(1,#,#,R,1)
(2,1,^,L,3)
(2,#,1,S,H)
(3,1,1,L,3)
(3,^,^,R,0)
(3,#,#,L,3)
(4,1,1,S,H)
(4,^,^,S,H)
(4,#,#,R,4)



Configurations (Sipser pg. 140)

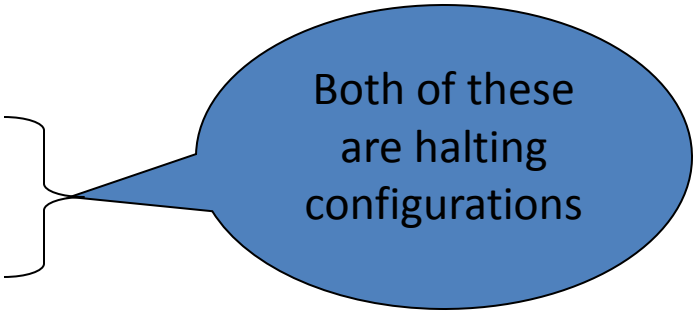
- configurations for TM's are strings of the form α q β , where $\alpha, \beta \in \Gamma^*$ and q $\in$ Q. (Assume that Q and Γ^* are disjoint sets, guaranteeing unique parsing of configurations.)
- The string α represents the tape contents to the left of the head.
- The string β represents the non-blank tape contents to the right of the head, including the currently scanned cell.

- Adding or deleting a few blank symbols at the beginning or end of an confiuration results in an equivalent configuartion. Both represent the same instant in the execution of a TM.

Sipser terminology

- Other authors call configurations instantaneous descriptions

- Starting Configuration
- Accepting Configuration
- Rejecting Configuration



Relating configurations

- TM's transitions induce the relation |- between configurations.
- Let $\omega = X_1 \ldots X_{i-1}\ q\ X_i \ldots X_k$ be an configuration.

- If $\delta(q,X_i)$ is undefined, then there are no configurations ω' such that $\omega \vdash \omega'$.

- If $\delta(q,X_i)=(p,Y,R)$ then
  $\omega \vdash \omega'$ holds for $\omega' = X_1 \ldots X_{i-1}\ Y\ p\ X_{i+1} \ldots X_k$

- Similarly, if $\delta(q,X\_i)=(p,Y,L)$
  then $\omega \vdash \omega'$ holds for $\omega' = X_1 \ldots pX_{i-1}YX_{i+1} \ldots X_k$

- When $\omega \vdash \omega'$ Sipser says: " ω yields ω' "

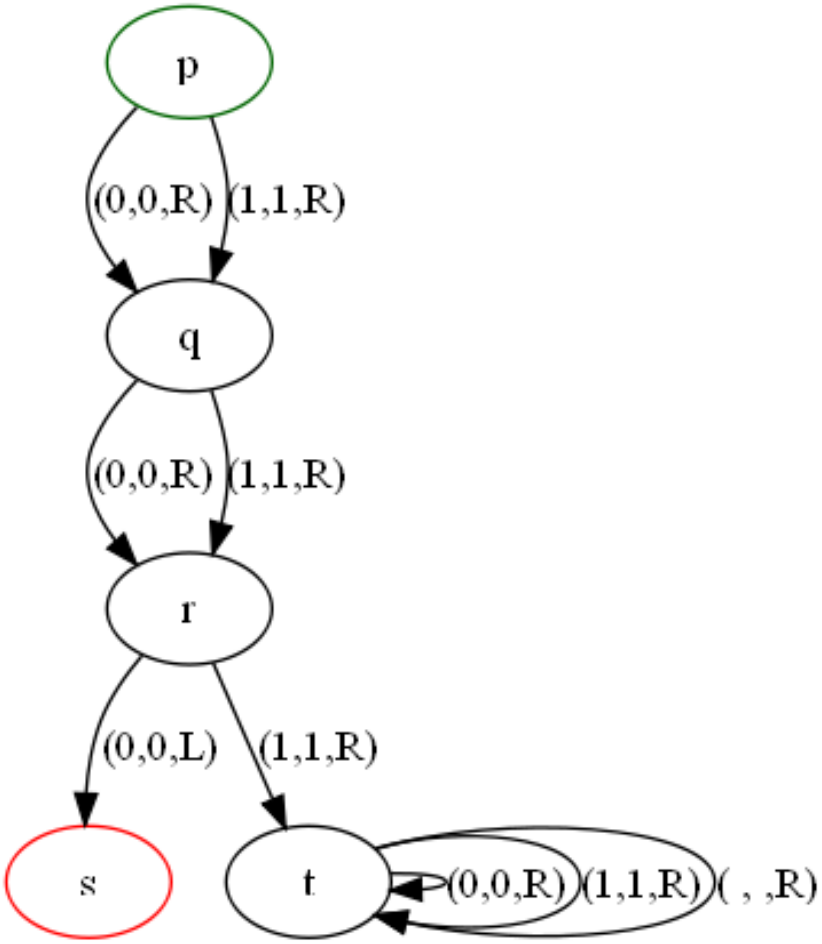
Note

- If, in the first case, we have i=k, (that is we are at the end of the non-blank portion of the tape to the right) then we need to use the equivalent representation

- $\omega = X_1 \ldots X_{k-1}\ q\ X_k\ B$

- for our formula to make sense. Similarly, we add a B to the beginning of ω whenever necessary.

Example



- Here is the sequence of configurations of our example machine, showing its execution with the given input 0101:
- p0101 |- 0q101 |-01r01 |- 0s101
- The machine halts, since there are no moves from the state s. When the input is 0111, the machine goes forever, as follows:
- p0111 |- 0q111 |- 01r11 |- 011t1 |- 0111t |- 0111Bt |- 0111BBt |- …

The Language of a TM

- We define the language of the TM M to be the set L(M) of all strings $w \in \Sigma^*$
- such that: $Q_0\, w \mid\!-^* \alpha\, q_{accept}\, \beta$
- for any α, β
- Languages accepted by TM's are call *recursively enumerable* (RE). Sipser calls this Turing-recognizable
- **Example**. For our example machine, we have L(M)= $(0+1)(0+1)0(0+1)^*$
- If the machine recognizes some language, and also halts for all inputs. We say the language is Turing-decidable.

Acceptance by Halting

- Some text books define an alternative way of defining a language associated with a TM M. (But not Sipser, though the idea is still interesting).
- We denote it H(M), and it consists of strings that cause the TM to halt. Precisely, a string $w \in \Sigma^*$ belongs to H(M)
- iff $q_0 w \vdash^* \alpha p X \beta$
- where $\delta(p,X)$ is undefined.

- **Example**. For our example machine, we have
- $H(M)= \varepsilon + 0 + 1 + (0+1)(0+1) + (0+1)(0+1)0(0+1)^*$

Equivalence of Acceptance by Final State and Halting

- How would we prove such an equivalence?

- 1. Construct a TM that accepts by Halting from an ordinary one.
- 2. Construct an ordinary TM from one that accepts by halting.