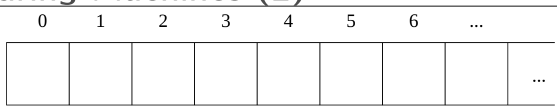
18 Turing Machines

1. Turing Machines
2. Nondeterministic Turing Machines

18.1 Turing Machines

- Introduced by Alan Turing
- Abstract machine with many features of computer
- Designed *before* the stored program computer!
- No memory limit!
- An infinite tape, and a tape reader, that reads one symbol at a time.

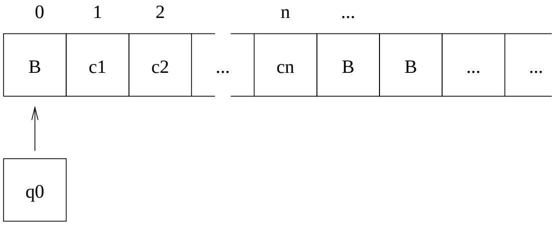
Turing Machines (2)



A quintuple $M = (Q, \Sigma, \Gamma, \delta, q0)$

- A finite set of states Q.
- A finite set Γ, the *tape alphabet*, which includes blank B
- A subset $\Sigma \subseteq \Gamma - \{B\}$ called the *input alphabet*
- A partial *transition function* $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
- A *start state* $q0 \in Q$

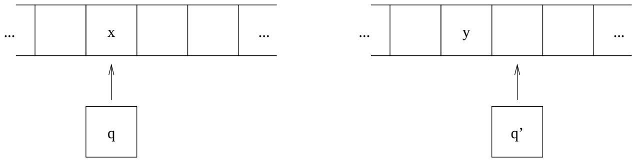
18.1.1 Turing Machines: Initial



Computation begins with

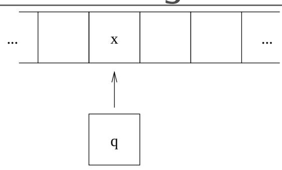
- An *input* string $s = c_1 \cdots c_n$ in Σ^* on the tape from position 1 to n
- The remaining tape blank (filled with B)
- Initial state $q0$

18.1.2 Turing Machines: Transition



- Machine in state q reading tape symbol $x \in \Gamma$
- $\delta(q, x) = [q', y, d]$
- Transitions have three parts:
  - Change the state to q'
  - Write symbol y on square scanned by tape head
  - Move head left or right (depending on direction d)

18.1.3 Turing Machines: Final



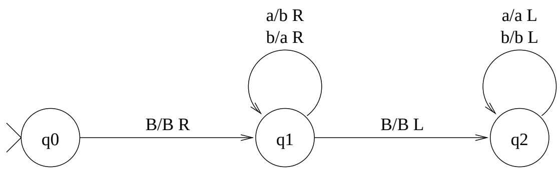
- Machine *halts* if in state q reading tape symbol $x \in \Gamma$ there is no $\delta(q, x) = [q', y, d]$
- Remember δ is a partial function.
- *Output* is the result remaining on the tape.
- Machine *halts abnormally* if the head moves off the left end of the tape.

18.1.4 Turing Machines: Example

Change all a's to b's and vice versa.

δ	B	a	b
$q0$	$q1, B, R$		
$q1$	$q2, B, L$	$q1, b, R$	$q1, a, R$
$q2$		$q2, a, L$	$q2, b, L$

Represented as a state machine

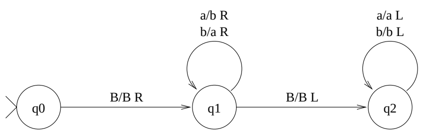


18.1.5 Turing Machines: Traces

- A *configuration* uq_ivB
  - represents the tape $uvBBB\ldots$ (v includes rightmost non blank)
  - shows tape head over first symbol in v
  - shows machine state q_i
- notation $uq_ivB \vdash xq_jyB$ indicates a transition from configuration uq_ivB to xq_jyB.
- $uq_ivB \vdash^* xq_jyB$ means result of any finite number of steps

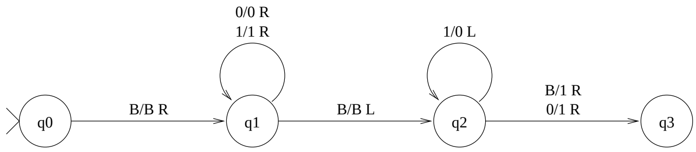
18.1.6 Turing Machines: Example Trace

$$
\begin{aligned}
& q_0BabaB \\
\vdash\ & Bq_1abaB \\
\vdash\ & Bbq_1baB \\
\vdash\ & Bbaq_1aB \\
\vdash\ & Bbabq_1B \\
\vdash\ & Bbaq_2bB \\
\vdash\ & Bbq_2abB \\
\vdash\ & Bq_2babB \\
\vdash\ & q_2BbabB
\end{aligned}
$$



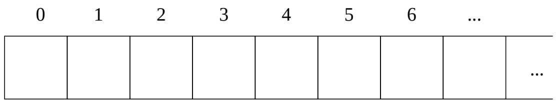
18.1.7 Turing Machines: Another Example

Machine to add 1 to a binary number



q_0B001B	q_0B11B
$\vdash Bq_1001B$	$\vdash Bq_111B$
$\vdash B0q_101B$	$\vdash B1q_11B$
$\vdash B00q_11B$	$\vdash B11q_1B$
$\vdash B001q_1B$	$\vdash B1q_21B$
$\vdash B00q_21B$	$\vdash Bq_210B$
$\vdash B0q_200B$	$\vdash q_2B00B$
$\vdash B01q_30B$	$\vdash 1q_300B$

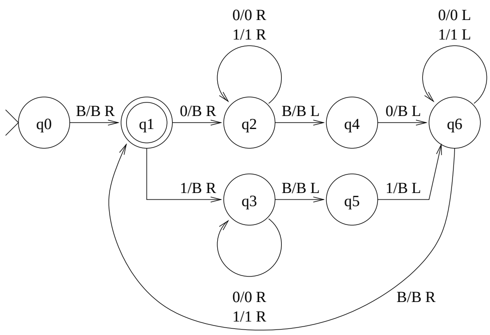
18.1.8 Turing Machines as Acceptors



A sextuple $M = (Q, \Sigma, \Gamma, \delta, q0, F)$

- F is the set of final states.
- If M halts in a state $q \in F$ then the input is accepted
- If M halts in a state $q \notin F$ the input is rejected.
- $L(M)$ is the set of strings accepted by M.

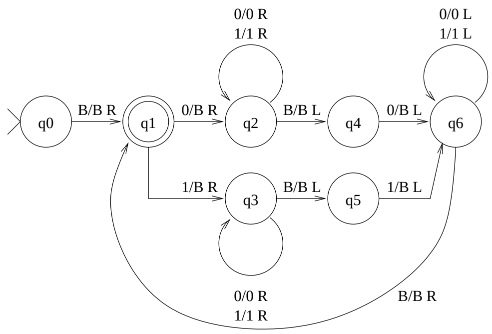
Turing Machines as Acceptors (2)



A machine that accepts a string which is an even length palindrome of 0's and 1's. e.g $00, 0110, 0001001000 \in L(M)$

Exercise

Modify the even length palendrome machine to accept odd length palindromes too e.g. 101



18.1.9 Turing Machines in Prolog

- given `delta(State,Sym,NewState,NewSym,LR)`
- (maybe) given `accepting(State)` for accepting states
- We can write a Turing machine simulator in Prolog
- So Prolog can do anything a Turing machine can

```
delta(q0, 'B', q1, 'B', right).
delta(q1, 'B', q2, 'B', left).
delta(q1, a, q1, b, right).
delta(q1, b, q1, a, right).
delta(q2, a, q2, a, left).
delta(q2, b, q2, b, left).

initial(q0).
```


Main Code

```
run(Input, Output, State) :-
        initial(State0),
        tape_list_position(Tape0, Input, 0),
        run(State0, Tape0, State, Tape),
        tape_list_position(Tape, Output, _).

recognize(Input) :-
        run(Input, _, State),
        accepting(State).
```

State Transition

```
run(State0, Tape0, State, Tape) :-
        (   step(State0, Tape0, State1, Tape1) ->
                run(State1, Tape1, State, Tape)
        ;   State = State0,
            Tape = Tape0
        ).

step(State0, Tape0, State, Tape) :-
        replace_tape_symbol(Tape0, Sym0, Tape1, Sym1),
        delta(State0, Sym0, State, Sym1, Direction),
        move_tape(Direction, Tape1, Tape).
```

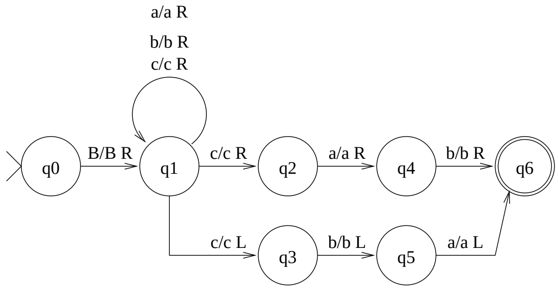

Handling the Tape

```
list_tape_position(List, tape(Left, Right), Pos) :-
        length(Left, Pos),
        reverse(Left, Left1),
        append(Left1, Right, List).

replace_tape_symbol(tape(Left0,Right0), Sym0, tape(Left0,Right), Sym) :-
        replace_next_of_infinite_tape(Right0, Sym0, Right, Sym).

move_tape(left,  tape([Lsym|Left],Right), tape(Left,[Lsym|Right])).
move_tape(right, tape(Left,Right), tape([Rsym|Left],Right1)) :-
        replace_next_of_infinite_tape(Right, Rsym, [_|Right1], Rsym).

replace_next_of_infinite_tape([Sym0|Rest], Sym0, [Sym|Rest], Sym).
replace_next_of_infinite_tape([], 'B', [Sym], Sym).
```


18.2 Nondeterministic Turing Machines

- δ maps to a subset of $Q \times \Gamma \times \{L, R\}$ rather than just one (or zero)
- Computation chooses which one to take?
- A non-deterministic Turing machine M accepts input s if some possible computation accepts s.
- **Important result:** Any language accepted by a non-deterministic Turing machine can be accepted by a deterministic Turing machine

Example



A machine which accepts strings of a's, b's and c's where there is a c followed by or preceded by ab.

Example (2)

$q_0 BacabB$	1	$q_0 BacabB$	1	$q_0 BacabB$	2
$\vdash Bq_1 acabB$	1	$\vdash Bq_1 acabB$	1	$\vdash Bq_1 acabB$	2
$\vdash Baq_1 cabB$	1	$\vdash Baq_1 cabB$	1	$\vdash Baq_1 cabB$	3
$\vdash Bacq_1 abB$	1	$\vdash Bacq_2 abB$	2	$\vdash Bq_3 acabB$	
$\vdash Bacaq_1 bB$	1	$\vdash Bacaq_4 bB$	1		
$\vdash Bacabq_1 B$		$\vdash Bacabq_6 B$			

First and third do not accept, but second does

That's good enough: the machine accepts