COMP2012/G52LAC Languages and Computation Lecture 15

Turing Machines

Henrik Nilsson

University of Nottingham

Turing Machines (1)

- A Turing Machine (TM) is a mathematical model of a general-purpose computer.

Turing Machines (1)

- A Turing Machine (TM) is a mathematical model of a general-purpose computer.
- A TM is a generalisation of a PDA: TM = FA + infinite tape

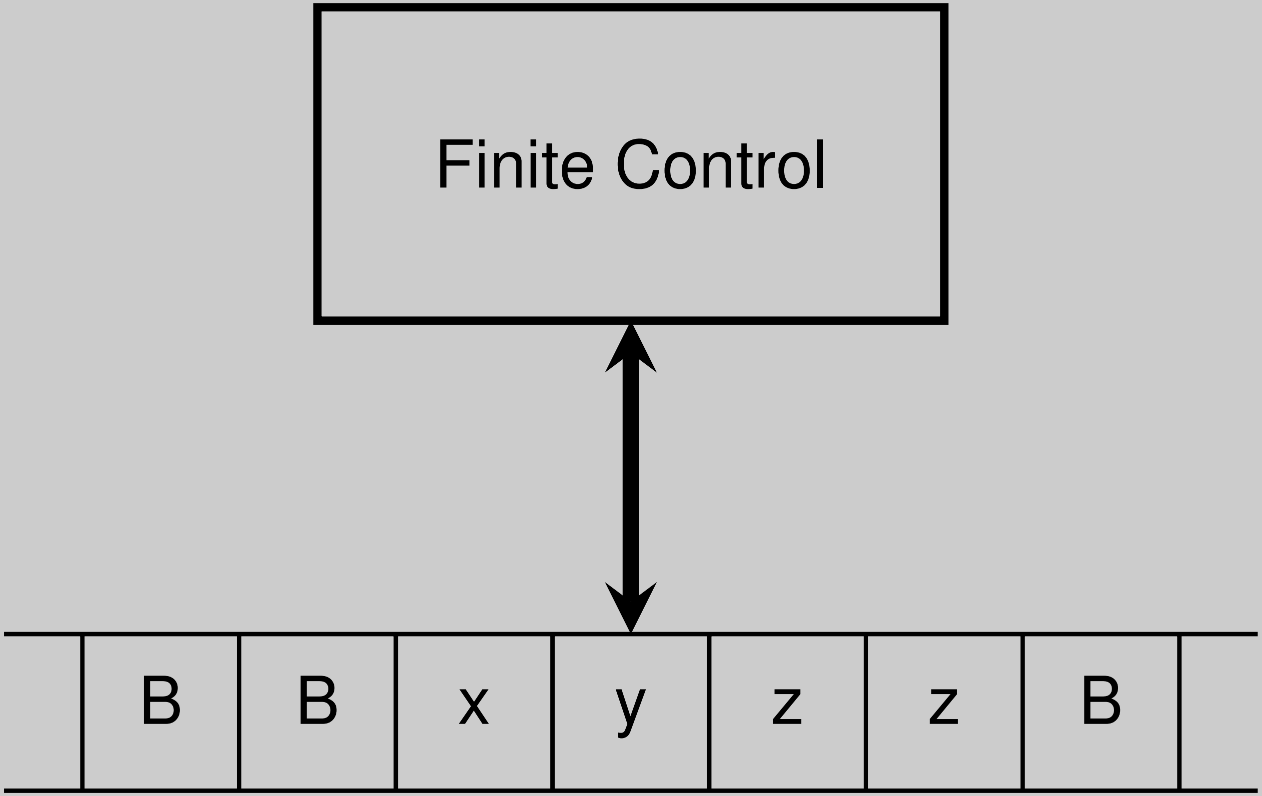
Turing Machines (1)

- A Turing Machine (TM) is a mathematical model of a general-purpose computer.
- A TM is a generalisation of a PDA: TM = FA + infinite tape
- Mainly used to study the ***notion of computation***: what (exactly!) can computers do (given sufficient time and memory) and what can they not do.

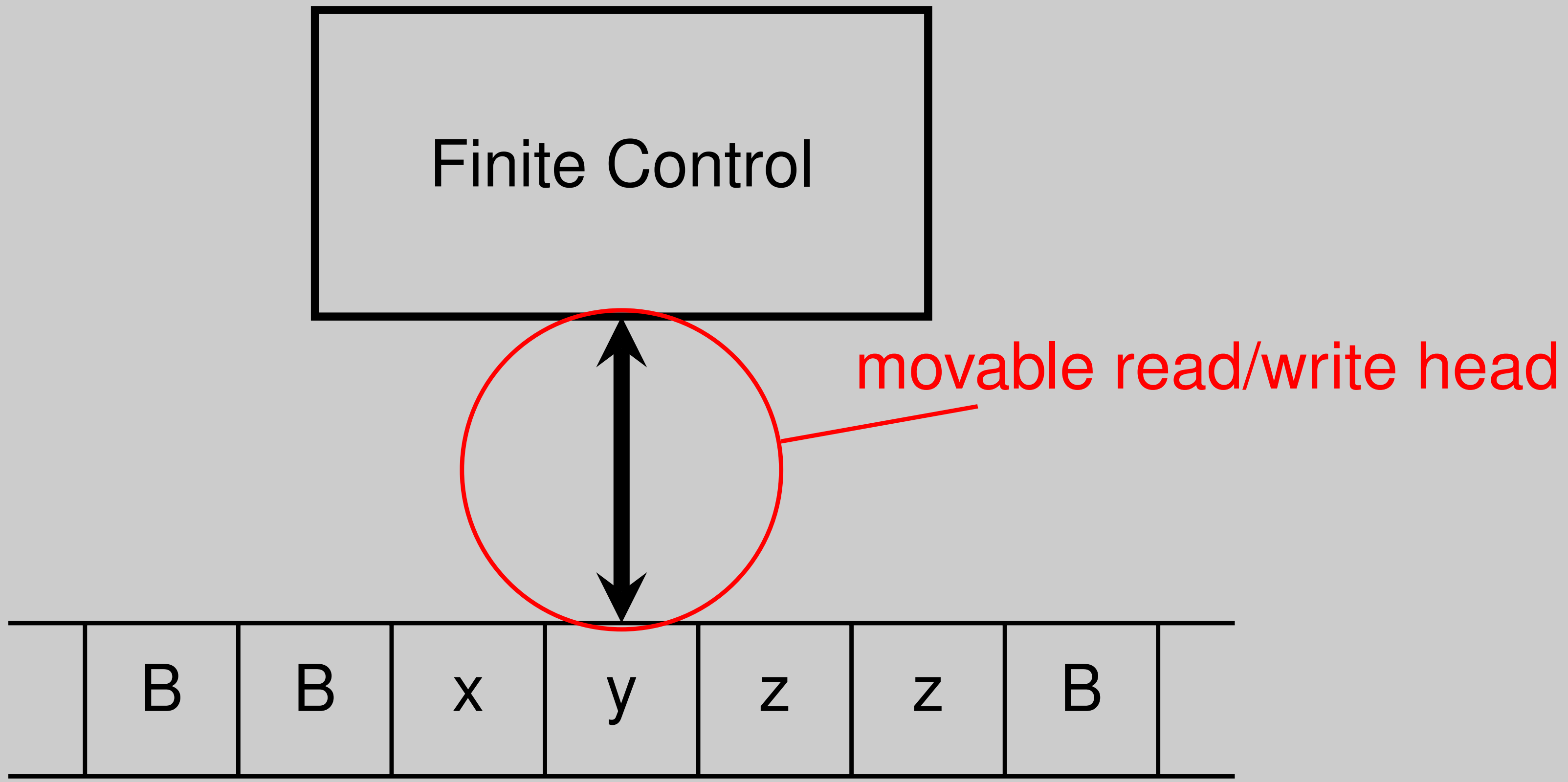
Turing Machines (1)

- A Turing Machine (TM) is a mathematical model of a general-purpose computer.
- A TM is a generalisation of a PDA: TM = FA + infinite tape
- Mainly used to study the ***notion of computation***: what (exactly!) can computers do (given sufficient time and memory) and what can they not do.
- There are other notions of computation, e.g. the λ***-calculus*** introduced by Alonzo Church (G54FOP!).

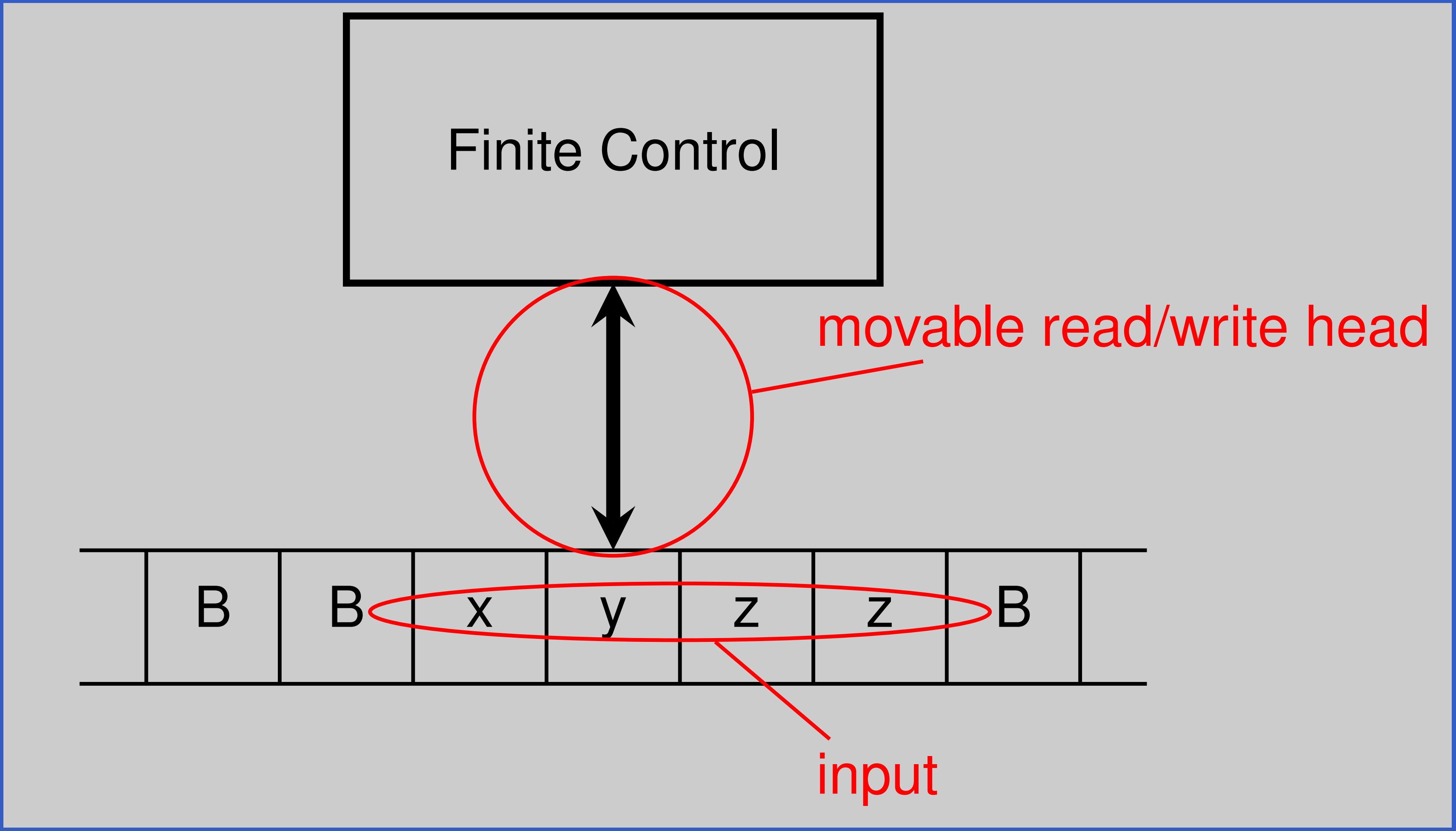
Turing Machines (2)



Turing Machines (2)



Turing Machines (2)



Turing Machines (3)

- All suggested notions of computation have so far proved to be equivalent.

Turing Machines (3)

- All suggested notions of computation have so far proved to be equivalent.
- ***The Church-Turing Thesis***: “Every function which would naturally be regarded as ‘computable’ can be computed by a TM”.

Turing Machines (3)

- All suggested notions of computation have so far proved to be equivalent.
- ***The Church-Turing Thesis***: “Every function which would naturally be regarded as ‘computable’ can be computed by a TM”.
- At first, given how simple TMs are, it may seem surprising they can do much at all. E.g. how can they even add or multiply?

Turing Machines (3)

- All suggested notions of computation have so far proved to be equivalent.
- ***The Church-Turing Thesis***: “Every function which would naturally be regarded as ‘computable’ can be computed by a TM”.
- At first, given how simple TMs are, it may seem surprising they can do much at all. E.g. how can they even add or multiply?
- We will see that a TM at least is more expressive than a PDA.

Definition of a Turing Machine

A TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where

- Q is a finite set of states
- Σ is the input alphabet
- Γ is the tape alphabet, $\Sigma \subset \Gamma$ (finite)
- $\delta \in Q \times \Gamma \to \{\mathrm{stop}\} \cup Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ is the transition function
- $q_0 \in Q$ is the initial state
- B is the blank symbol, $B \in \Gamma$, $B \notin \Sigma$
- $F \subseteq Q$ are the accepting (final) states

Instantaneous Description (ID)

Instantaneous Descriptions (ID) describe the ***state*** of a TM computation:

$$ID = \Gamma^* \times Q \times \Gamma^*$$

$(\gamma_L, q, \gamma_R) \in ID$ means:

- TM is in state q
- γ_L is the non-blank part of the tape to the ***left*** of the head.
- γ_R is the non-blank part of the tape to the ***right*** of the head, ***including*** the current position.

The Next State Relation (1)

The next state relation on ID:

$$\underset{M}{\vdash} \subseteq ID \times ID$$

Read

$$id_1 \underset{M}{\vdash} id_2$$

"TM M moves in one step from id_1 to id_2."

The Next State Relation (2)

Let $q, q' \in Q$, $x, y, z \in \Gamma$, $\gamma_L, \gamma_R \in \Gamma^*$

1. $(\gamma_L, q, x\gamma_R) \underset{M}{\vdash} (\gamma_L y, q', \gamma_R)$ if $\delta(q, x) = (q', y, R)$
2. $(\gamma_L z, q, x\gamma_R) \underset{M}{\vdash} (\gamma_L, q', zy\gamma_R)$ if $\delta(q, x) = (q', y, L)$
3. $(\epsilon, q, x\gamma_R) \underset{M}{\vdash} (\epsilon, q', By\gamma_R)$ if $\delta(q, x) = (q', y, L)$
4. $(\gamma_L, q, \epsilon) \underset{M}{\vdash} (\gamma_L y, q', \epsilon)$ if $\delta(q, B) = (q', y, R)$
5. $(\gamma_L z, q, \epsilon) \underset{M}{\vdash} (\gamma_L, q', zy)$ if $\delta(q, B) = (q', y, L)$
6. $(\epsilon, q, \epsilon) \underset{M}{\vdash} (\epsilon, q', By)$ if $\delta(q, B) = (q', y, L)$

The Language of a TM (1)

$$L(M) = \{w \in \Sigma^* \mid (\epsilon, q_0, w) \overset{*}{\underset{M}{\vdash}} (\gamma_L, q, \gamma_R) \ \wedge \ q \in F\}$$

The Language of a TM (1)

$$L(M) = \{w \in \Sigma^* \mid (\epsilon, q_0, w) \overset{*}{\underset{M}{\vdash}} (\gamma_L, q, \gamma_R) \ \wedge \ q \in F\}$$

A TM stops if it reaches an accepting state.

The Language of a TM (1)

$$L(M) = \{w \in \Sigma^* \mid (\epsilon, q_0, w) \overset{*}{\underset{M}{\vdash}} (\gamma_L, q, \gamma_R) \ \wedge \ q \in F\}$$

A TM stops if it reaches an accepting state.

A TM stops in a non-accepting state if the transition function returns stop for that state and current tape input.

The Language of a TM (1)

$$L(M) = \{w \in \Sigma^* \mid (\epsilon, q_0, w) \overset{*}{\underset{M}{\vdash}} (\gamma_L, q, \gamma_R) \ \wedge \ q \in F\}$$

A TM stops if it reaches an accepting state.

A TM stops in a non-accepting state if the transition function returns stop for that state and current tape input.

However, it may also ***never*** stop!

This is unlike the machines we have encountered before.

The Language of a TM (2)

If a particular TM M ***always*** stops, either in an accepting or a non-accepting state, then M ***decides*** $L(M)$.

The Language of a TM (2)

If a particular TM M ***always*** stops, either in an accepting or a non-accepting state, then M ***decides*** $L(M)$.

Given that TMs model general purpose computers, it should not come as a surprise that they can loop. Consider e.g.

```
input x; while (x<10);
```

The Language of a TM (2)

If a particular TM M ***always*** stops, either in an accepting or a non-accepting state, then M ***decides*** $L(M)$.

Given that TMs model general purpose computers, it should not come as a surprise that they can loop. Consider e.g.

```
input x; while (x<10);
```

What may come as a surprise is that there are languages for which a TM ***necessarily*** cannot decide membership; i.e., will loop on some inputs.

Example

Construct a TM that accepts the language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$.

This is a language that cannot be defined by a CFG or recognized by a PDA.

On the whiteboard.

There are many TM similators on-line. Try this (or some other) example with one of those. E.g.:

`http://ironphoenix.org/tm`