Turing Machines

A Model of Computation

Alan Turing

- Born in 1912
- 1922: Troubles in School
- 1936: Turing Machine
- WWII: Bletchley Park
- 1954: Suicide?!

Original Motivation

Modeling "Human Computers"

What are "Human Computers"?

Remember the Turing Test!

Original Motivation

Modeling "Human Computers"

What are "Human Computers"?

(might be a biased picture)

Today's Perspective

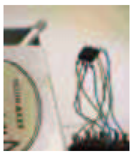
Modeling "Human Computers"

What are "Human Computers"?

Today's Perspective

Modeling "Computers"

What are "Computers"?



Today's Perspective

Modeling "Computers"

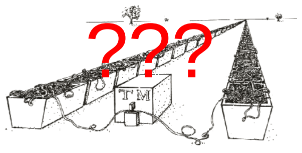
What are "Computers"?

- Storage Device
  - read/write access
  - finite size (conceptually arbitrarily large)
- Control Unit
  - defines which step to do next
  - aka CPUs/Programs

Today's Perspective

Modeling "Computers"

What are "Computers"?

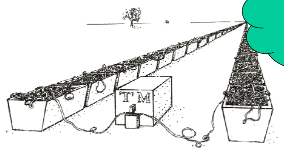


Today's Perspective

Modeling "Computers"

What are "Computers"?

Modeling…
But what
for???



Today's Perspective

Modeling "Computers"

What are "Computers"?

Modeling…
But what
for???

Reasoning about
- algorithms
- computational problems

Treating them as Mathematical Objects!!!

Algorithms as Mathematical Objects

- TMs are meant for formulating & proving general stat ments ut orithms:
  – What is co
  – How m ed to so
- TMs ar
  – program g
  – real computations
  – browsing the web

So what?!
Turing Machines
and real computers?!

Relevance of TMs: What is Computable by TMs?

LCMs can do anything that could be described as "rule of thumb" or "purely mechanical".

This is sufficiently well established that it is now agreed amongst logicians that "calculable by means of an LCM" is the correct accurate rendering of such phrases.

(Turing in 1948 on his Logical Computing Machine)

Relevance of TMs: Church-Turing Thesis

LCMs can do anything that could be described as "rule of thumb" or "purely mechanical".

- Historically: A lot of models are equivalent to TMs (i.e., they describe the same set of algorithms)
  - Lambda Calculus
  - Partially Recursive Functions
  - …
- Practically: All known computer systems are equivalent to TMs.

Relevance of TMs: How efficient are TMs?

All reasonable models of computation are polynomially related to the TM wrt. their time performance.

This is established by simulation arguments…

Relevance of TMs: How efficient are TMs?

All reasonable models of computation are polynomially related to the TM wrt. their time performance.

This is established by simulation arguments…

A Pentium 4 can be simulated by a TM with runtime n^k for some fixed k.

Relevance of TMs: Extended Church-Turing Thesis

All reasonable models of computation are polynomially related to the TM wrt. their time performance.

This is established by simulation arguments…
…. it's a thesis – not a theorem.

DNA-Computing

Quantum-Computing

Relevance of TMs: Extended Church-Turing Thesis

TMs can simulate real computers efficiently

TMs have a mathematically simple structure

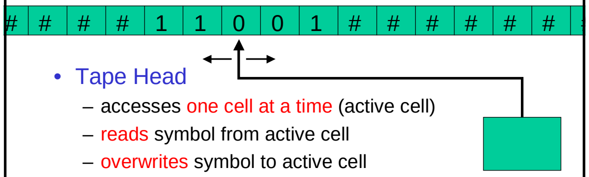
TMs are the ideal vehicle to build a Theory on Efficient Computability

This Lecture

- Definition of TMs
- Execution of TMs
- Multi-Tape TMs
- Non-Deterministic TMs

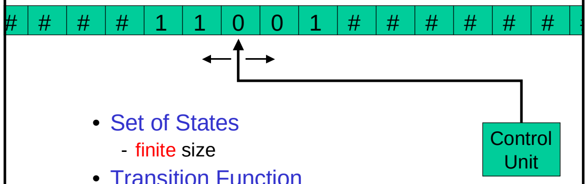
Storage Device of a TM

- Tape
  - arbitrarily long but finite strip divided into cells
  - each cell contains a single symbol
  - finite alphabet of symbols



- Tape Head
  - accesses one cell at a time (active cell)
  - reads symbol from active cell
  - overwrites symbol to active cell
  - moves left, right or stays

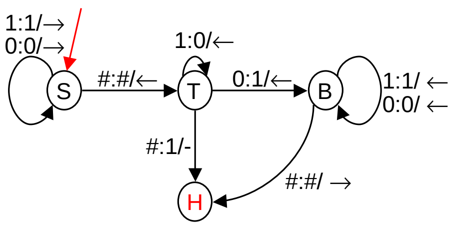
Control Unit of a TM



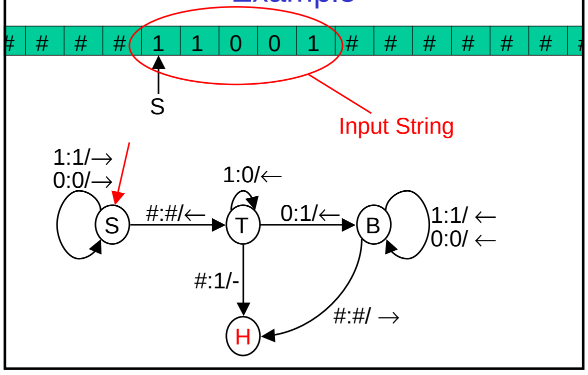
- Set of States
  - finite size
- Transition Function
  given a state and an input symbol, the control decides which
  - symbol to write
  - direction the head is moved
  - new state to assume

Example

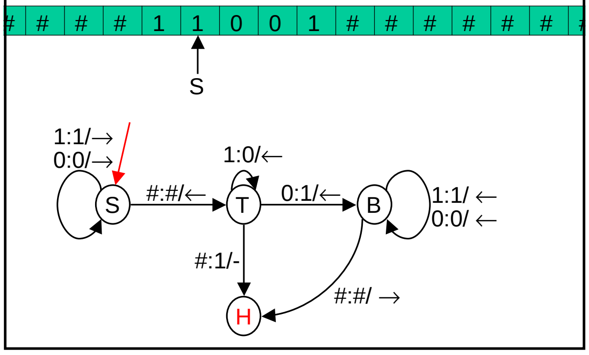
Alphabet: {0,1}
Blank Symbol: #



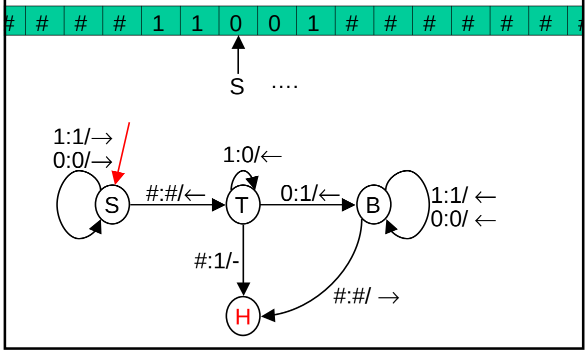
Example



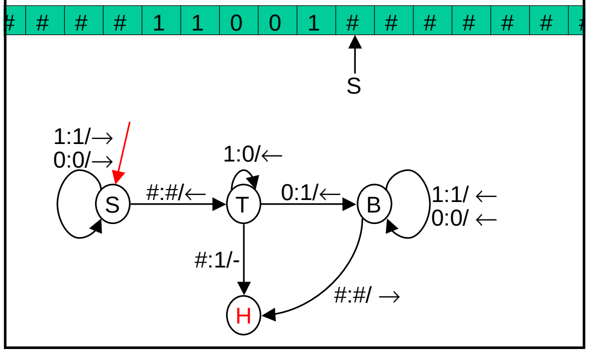
Example



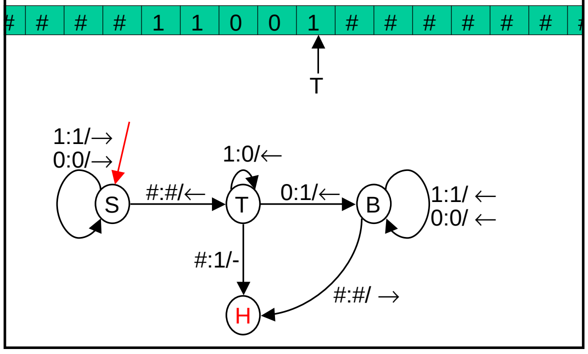
Example



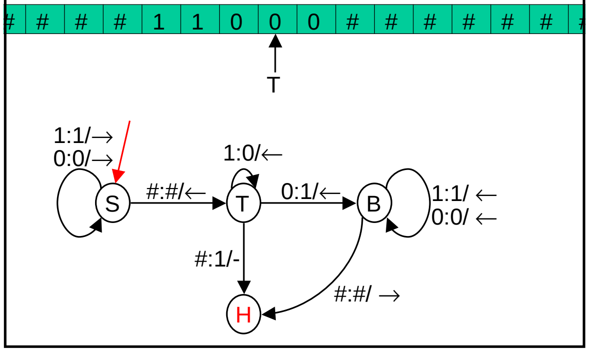
Example



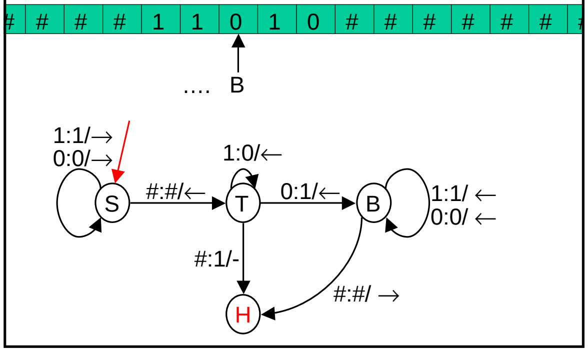
Example



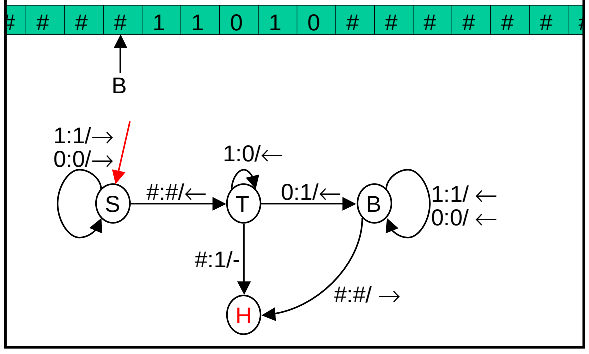
Example



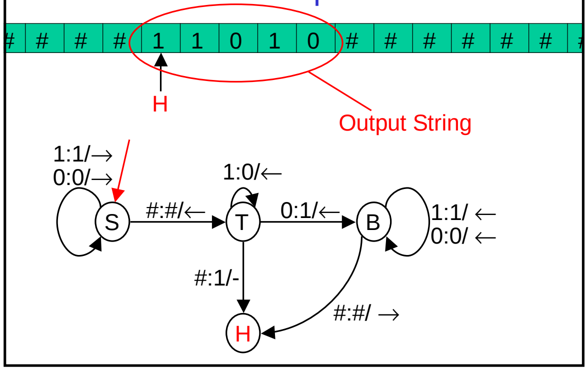
Example



Example



Example



Formal Turing Machine Definition

$M = \langle K, \Sigma, \delta, s \rangle$

- finite set of states K

- finite set of symboles Σ (alphabet)

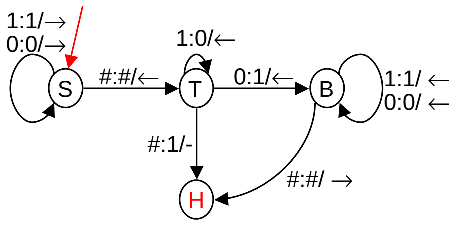
- transition function
  $\delta : K \times \Sigma \to (K \cup \{\mathrm{H}, \mathrm{A}, \mathrm{R}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$

- initial state $s \in K$

Example

Alphabet: {0,1}
Blank Symbol: #

$M = \langle K, \Sigma, \delta, s \rangle$

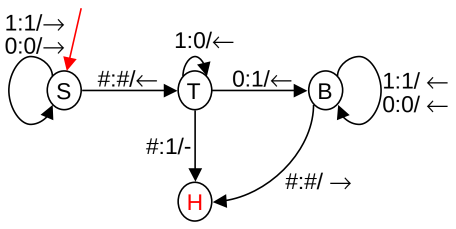


Example

# is always included

$\Sigma = \{0,1,\#\}$

$M = \langle K, \Sigma, \delta, s \rangle$

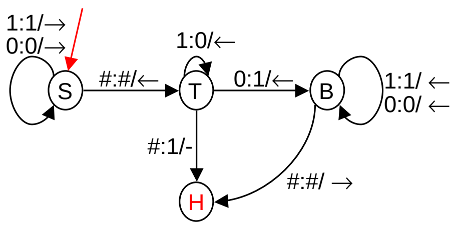


Example

State H is never left!

$\Sigma = \{0,1,\#\}$
$K = \{\mathrm{S}, \mathrm{T}, \mathrm{B}\}$

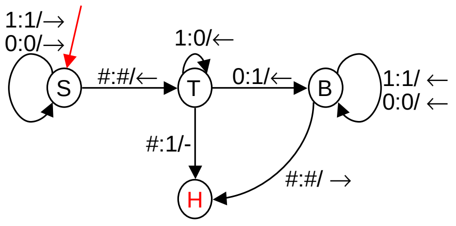
$M = \langle K, \Sigma, \delta, s \rangle$



Example

$\Sigma = \{0,1,\#\}$
$K = \{\mathrm{S}, \mathrm{T}, \mathrm{B}\}$
$s = S$

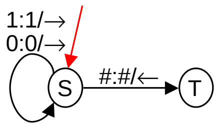
$M = \langle K, \Sigma, \delta, s \rangle$



Example

$\Sigma = \{0,1,\#\}$
$K = \{\mathrm{S}, \mathrm{T}, \mathrm{B}\}$
$s = S$

$M = \langle K, \Sigma, \delta, s \rangle$



$\delta(\mathrm{S},1) = < S, 1, \rightarrow >$
$\delta(\mathrm{S},0) = < S, 0, \rightarrow >$
$\delta(S,\#) = < T, \#, \leftarrow >$
....

This Lecture

- Definition of TMs
- Execution of TMs
- Multi-Tape TMs
- Non-Deterministic TMs

Formal Turing Machine Definition

$M = \langle K, \Sigma, \delta, s \rangle$

- finite set of states K

- finite set of symboles Σ (alphabet)

- transition function
$\delta : K \times \Sigma \rightarrow (K \cup \{\mathrm{H}, \mathrm{A}, \mathrm{R}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$

- initial state $s \in K$

This Lecture

- Definition of TMs
- Execution of TMs
- Multi-Tape TMs
- Non-Deterministic TMs

Execution of TMs

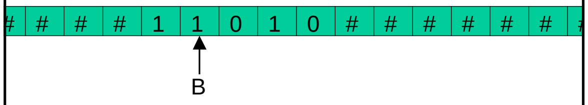
The execution of a TM is described formally as a Sequence of Configurations.

A Step of TM is the transition from one Configuration to the next one.

Two special configurations:
- Initial Configuration
- Halting Configuration

Configuration

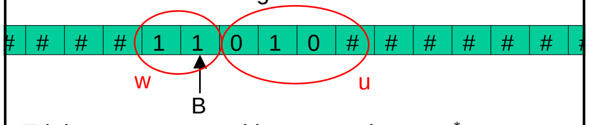
A Configuration C of $M = \langle K, \Sigma, \delta, s \rangle$ describes the entire state of M during some execution.



It must contain:
- cursor position
- tape contents
- state

Configuration

A Configuration C of $M = \langle K, \Sigma, \delta, s \rangle$ describes the entire state of M during some execution.

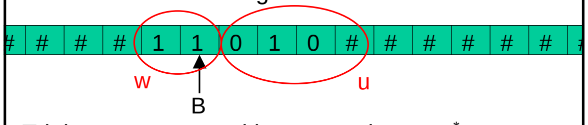


Triple $C =< q, w, u >$ with $q \in K$ and $w, u \in \Sigma^*$

w is the string up until the tape head
u contains the rest
#s which have not been visited are ignored

Configuration

A Configuration C of $M = \langle K, \Sigma, \delta, s \rangle$ describes the entire state of M during some execution.



Triple $C =< q, w, u >$ with $q \in K$ and $w, u \in \Sigma^*$

$C =< \mathrm{B}, 11, 010\# >$

A Computational Step

$C = < q, w, u >$ and $\delta(q, w_n) = < q', w_n', dir >$.

If $dir = \rightarrow$ then $w' = w_1 \ldots w_{n-1} w_n' u_1$

$u' = u_2 \ldots u_n$

#s are padded

A Computational Step

$C = < q, w, u >$ and $\delta(q, w_n) = < q', w_n', dir >$.

If $dir = \rightarrow$ then $w' = w_1 \ldots w_{n-1} w_n' u_1$

$u' = u_2 \ldots u_n$

#s are padded

For $dir = \rightarrow$ and $dir = -$ analogously.

C is said to yield $C' = < q', w', u' >$.

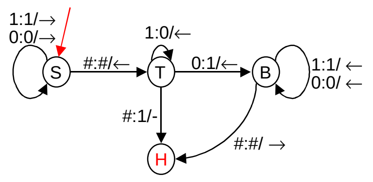
The transition from C to C' is a single step.

Example

#	#	#	#	1	1	0	0	1	#	#	#	#	#	#	#

$\uparrow$ S

$C = < \text{S}, 1, 1001 >$

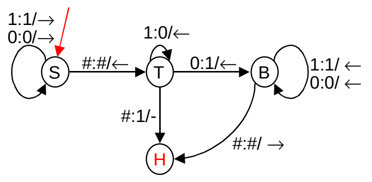


Example

#	#	#	#	1	1	0	0	1	#	#	#	#	#	#	#

$\uparrow$ S …

$C = < \text{S}, 1, 1001 >$
$C = < \text{S}, 11, 001 >$
…

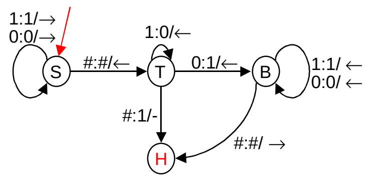


Example

#	#	#	#	1	1	0	0	1	#	#	#	#	#	#	#

$\uparrow$ S

$C = < \text{S}, 1, 1001 >$
$C = < \text{S}, 11, 001 >$
…
$C = < \text{S}, 11001\#, \varepsilon >$

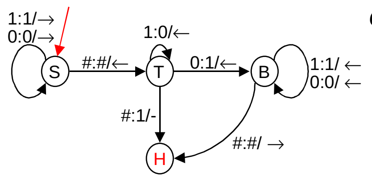


Example

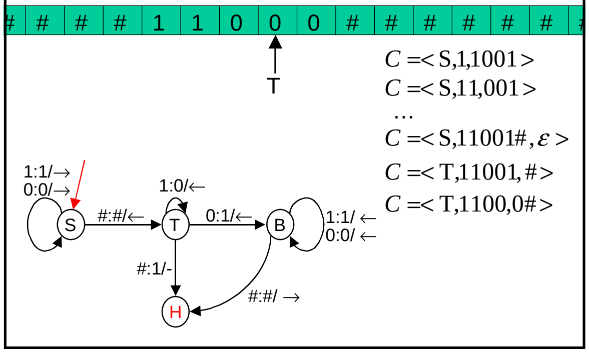
#	#	#	#	1	1	0	0	1	#	#	#	#	#	#	#

$\uparrow$ T

$C = < \text{S}, 1, 1001 >$
$C = < \text{S}, 11, 001 >$
…
$C = < \text{S}, 11001\#, \varepsilon >$
$C = < \text{T}, 11001, \# >$

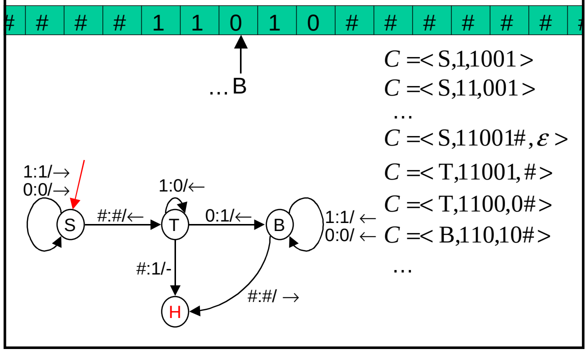


Example



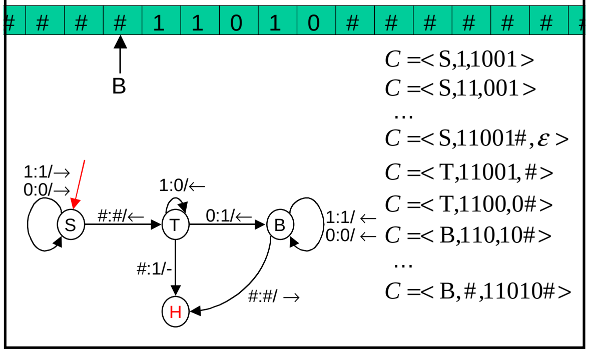
$C =< \mathrm{S,1,1001} >$
$C =< \mathrm{S,11,001} >$
…
$C =< \mathrm{S,11001\#}, \varepsilon >$
$C =< \mathrm{T,11001,\#} >$
$C =< \mathrm{T,1100,0\#} >$

Example



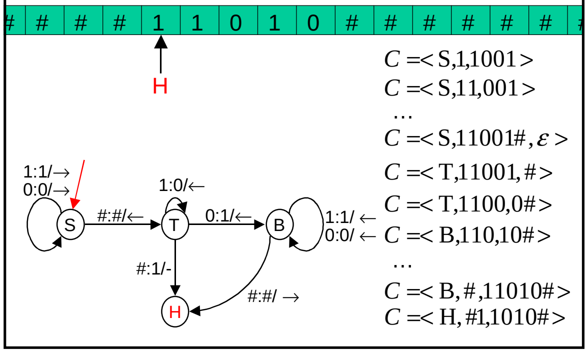
$C =< \mathrm{S,1,1001} >$
$C =< \mathrm{S,11,001} >$
…
$C =< \mathrm{S,11001\#}, \varepsilon >$
$C =< \mathrm{T,11001,\#} >$
$C =< \mathrm{T,1100,0\#} >$
$C =< \mathrm{B,110,10\#} >$
…

Example



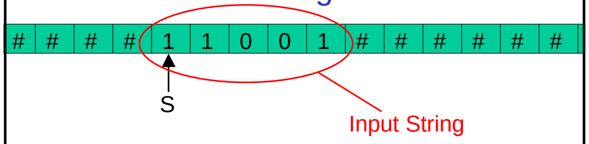
$C =< \mathrm{S,1,1001} >$
$C =< \mathrm{S,11,001} >$
…
$C =< \mathrm{S,11001\#}, \varepsilon >$
$C =< \mathrm{T,11001,\#} >$
$C =< \mathrm{T,1100,0\#} >$
$C =< \mathrm{B,110,10\#} >$
…
$C =< \mathrm{B,\#,11010\#} >$

Example



$C =< \mathrm{S,1,1001} >$
$C =< \mathrm{S,11,001} >$
…
$C =< \mathrm{S,11001\#}, \varepsilon >$
$C =< \mathrm{T,11001,\#} >$
$C =< \mathrm{T,1100,0\#} >$
$C =< \mathrm{B,110,10\#} >$
…
$C =< \mathrm{B,\#,11010\#} >$
$C =< \mathrm{H,\#1,1010\#} >$

Initial Configuration



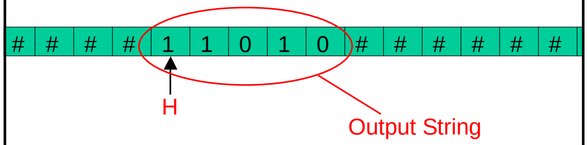
In our example, we started with the configuration

$C =< S,1,1001 >$

Given $M = \langle K, \Sigma, \delta, s \rangle$ and input $x \in (\Sigma - \{\#\})^*$

$C =< s, x_1, x_2 \ldots x_n >$ is the initial configuration

Halting Configuration



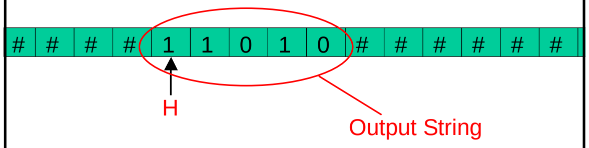
In our example, we halt with the configuration

$C =< H,\#1,1010\# >$

$C =< q, w, u >$ is a halting configuration, if

$q \in \{H, A, R\}$

Halting Configuration: Functions



If q=H then the M computed a function.

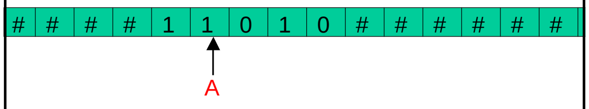
The result is the string between the head and the first # to the right.

In our example: $M(x) = x+1$

Excursus: Decision Problems

- Remember the UNO-Problem.
- Given a set of states, you can ask whether there is peaceful seating arrangement.
- This a decision problem: The answer is a single bit.
- Their simple structure is helpful within complexity.
- Formal Language: The set of positive instances.
- Functional problems can be “reduced” to decision problems.

Halting Configuration: Decision



If q=A (q=R) then M accepted (rejected) the input x.

Such an $M = < K, \Sigma, \delta, s >$ decides a language:
$L(M) = \{x \in \Sigma^* : M(x) = \text{A}\}$

Runtime of a TM

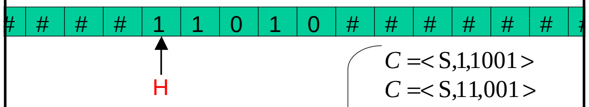
Let $M = \langle K, \Sigma, \delta, s \rangle$ and $x \in (\Sigma - \{\#\})^*$.

The the number of steps between initial and halting configuration is the runtime of M on x.

$\boxed{\text{If } M \text{ halts within } f(|x|) \text{ steps or less for all } x \in (\Sigma-\{\#\})^* \text{ then } M \text{ runs in time } f(n).}$

If M does not reach a halting state (H,A,R), then M does not terminate (runs forever).

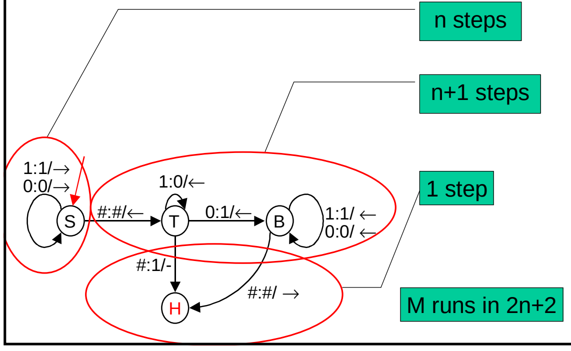
Example



Runtime of M on 110110: 12 steps

$$
\left.\begin{array}{l}
C = < \text{S,1,1001} > \\
C = < \text{S,11,001} > \\
\ \dots \\
C = < \text{S,11001\#}, \varepsilon > \\
C = < \text{T,11001,\#} > \\
C = < \text{T,1100,0\#} > \\
C = < \text{B,110,10\#} > \\
\ \dots \\
C = < \text{B,\#,11010\#} > \\
C = < \text{H,\#1,1010\#} >
\end{array}\right\}
$$

Example



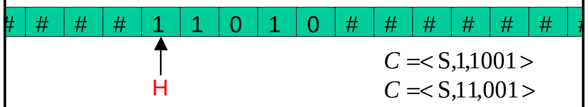
Space used by a TM

Let $M = \langle K, \Sigma, \delta, s \rangle$ and $x \in (\Sigma - \{\#\})^*$.

The number of symbols in the largest configuration is the space required by M on input x.

If M runs within $f(|x|)$ space or less for all $x \in (\Sigma - \{\#\})^*$ then M runs in space $f(n)$.

Example



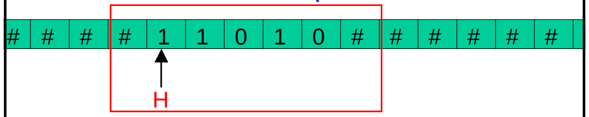
$C =< \mathrm{S}, 1, 1001 >$
$C =< \mathrm{S}, 11, 001 >$
…
$C =< \mathrm{S}, 11001\#, \varepsilon >$
$C =< \mathrm{T}, 11001, \# >$
$C =< \mathrm{T}, 1100, 0\# >$
$C =< \mathrm{B}, 110, 10\# >$
…

Space requirement of M on input 110110:
7 cells

Maximal configurations

$C =< \mathrm{B}, \#, 11010\# >$
$C =< \mathrm{H}, \#1, 1010\# >$

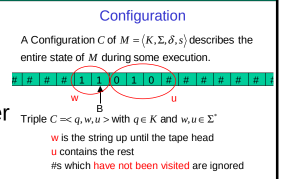
Example



M runs in space n+2

This Lecture

- Definition of TMs
- Execution of TMs
- Multi-Tape TMs
- Non-Deterministic TMs
- Encoding
- Constants do not matter



Configuration

A Configuration C of $M = \langle K, \Sigma, \delta, s \rangle$ describes the entire state of M during some execution.

Triple $C =< q, w, u >$ with $q \in K$ and $w, u \in \Sigma^*$

w is the string up until the tape head
u contains the rest
#s which have not been visited are ignored

This Lecture

- Definition of TMs
- Execution of TMs
- Multi-Tape TMs
- Non-Deterministic TMs
- Encoding
- Constants do not matter

Multi-Tape TMs

- Instead of a single tape, we use several tapes
- They are dedicated:
  - Input Tape (read only)
  - Work Tapes (read/write)
  - Output Tape (write only)

Multi-Tape TMs: Definition

adapting the definition $M = \langle K, \Sigma, \delta, s \rangle$:

If M is a k tape TM, then

$\delta : K \times \Sigma^k \rightarrow (K \cup \{\mathrm{H}, \mathrm{A}, \mathrm{R}\}) \times \Sigma^k \times \{\leftarrow, \rightarrow, -\}^k$

Read and write k symbols,
move on k tapes

rest is the same.

Multi-Tape TMs: Configuration

adpating $C = < q, w, u >$ with $q \in K$ and $w, u \in \Sigma^*$:

If M is a k tape TM, then

$C = < q, w_1, u_1, \ldots w_k, u_k >$ with $q \in K$ and $w_i, u_i \in \Sigma^*$

…. just k tapes

Multi-Tape TMs: Space Bound

$C = < q, w_1, u_1, \ldots w_k, u_k >$ with $q \in K$ and $w_i, u_i \in \Sigma^*$

The number of symbols in the largest configuration is the space required by M on input x.

But only the contents of the work tapes are counted!

I.e., input and output are not considered for space bounds.

Multi-Tape TMs: Stronger??

Let M be a k – tape TM running in time $O(f(n))$.
Then there is a 1- tape TM M' with $M(x) = M'(x)$
which runs in time $O(f^2(n))$.

(On the other hand: Palindroms can be decided by a
- 2-tape TM within time O(n)
- 1-tape TM requires O(n^2).)

This Lecture

- Definition of TMs
- Execution of TMs
- Multi-Tape TMs
- Non-Deterministic TMs

Multi-Tape TMs

- Instead of a single tape, we use several tapes
- They are dedicated:
  - Input Tape (read only)
  - Work Tapes (read/write)
  - Output Tape (write only)

This Lecture

- Definition of TMs
- Execution of TMs
- Multi-Tape TMs
- Non-Deterministic TMs

Deterministic TMs

The TMs we saw so far were deterministic.

I.e., the input determined the outcome of the computation.

I.e., we used a transition function:

$\delta : K \times \Sigma \rightarrow (K \cup \{\mathrm{H}, \mathrm{A}, \mathrm{R}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$

That's the way, our real computers work….

Non-Deterministic TMs

Non-Deterministic TMs are a formalism to express certain algorithms.

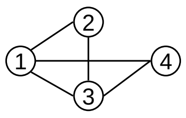
…. but you cannot simulate a nondet. TM directly by a real computer…

We start with an example…

Example: UNO

Given an undirected graph $G = < V, E >$.

Is there a circle which includes all nodes in V ?



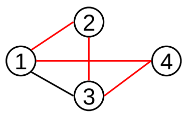
Note: We are only looking at the decision problem …

Example: UNO

Given an undirected graph $G = < V, E >$.

Is there a circle which includes all nodes in V ?

Sure:



Example: UNO

Given an undirected graph $G = < V, E >$.

Is there a circle which includes all nodes in V ?

If you try to solve this problem, you will end up enumerating the possible solutions…

```
1. for each permutation $\pi$ of $V$
2.    if $\pi$ is a path in $G$ then accept
3. reject
```

Example: UNO

```
1. for each perm[illegible] $V$
2.    if $\pi$ is [illegible] cept
3. reject
```

It might make a wrong guess?!?

Almost equivalently, [illegible]u can write:

```
1. guess a permutation $\pi$ of $V$
2. if $\pi$ is a path in $G$ then accept
3. reject
```

Example: UNO

1. guess a permutation π of V
2. if π is a path in G then accept
3. reject

… it might make a wrong guess, but

if there exists a solution, at least one guess will find it!

A way of capture such “algorithms”:
Non-deterministic TMs.

Deterministic TMs

We emphasized the fact that det. TMs use a transition function.

$$\delta : K\times\Sigma \to (K\cup\{\mathrm{H},\mathrm{A},\mathrm{R}\})\times\Sigma\times\{\leftarrow,\to,-\}$$

Non-Deterministic and Deterministic TMs

We emphasized the fact that det. TMs use a transition function.

$$\delta : K\times\Sigma \to (K\cup\{\mathrm{H},\mathrm{A},\mathrm{R}\})\times\Sigma\times\{\leftarrow,\to,-\}$$

For a reason. Non-det. TMs use a transition relation.

$$\delta \subseteq K\times\Sigma\times(K\cup\{\mathrm{H},\mathrm{A},\mathrm{R}\})\times\Sigma\times\{\leftarrow,\to,-\}$$

Non-Deterministic TMs

For a reason. Non-det. TMs use a transition relation.

$$\delta \subseteq K\times\Sigma\times(K\cup\{\mathrm{H},\mathrm{A},\mathrm{R}\})\times\Sigma\times\{\leftarrow,\to,-\}$$

Configurations are still the same :

$C = <q, w, u>$ with $q \in K$ and $w, u \in \Sigma^*$

But how does it run???

A Computational Step

$C = <q, w, u>$ and $\delta(q, w_n) = <q', w_n', dir>$.

If $dir = \to$ then $w' = w_1 \dots w_{n-1} w_n' u_1$

$u' = u_2 \dots u_n$

For $dir = \to$ and $dir = -$ analogously.

C is said to yield $C' = <q', w', u'>$.

The transition from C to C’ is a single step.

A Nondeterministic Computational Step

$C = <q, w, u>$ and $<q, w_n, q', w_n', dir> \in \delta$.

If $dir = \to$ then $w' = w_1 \dots w_{n-1} w_n' u_1$

$u' = u_2 \dots u_n$

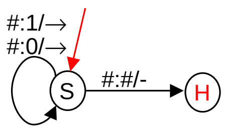
For $dir = \to$ and $dir = -$ analogously.

C is said to yield $C' = <q', w', u'>$.

The transition from C to C’ is a single step.

Example

$C = < \mathrm{S}, \#, \varepsilon >$



Example

$$\begin{tikzcd}
 & C = < \mathrm{S}, \#, \varepsilon > \arrow[dl] \arrow[d] \arrow[dr] & \\
C = < \mathrm{S}, 1\#, \varepsilon > & C = < \mathrm{S}, 0\#, \varepsilon > & C = < \mathrm{H}, \#, \varepsilon >
\end{tikzcd}$$