# Sec. 4: Turing Machines

(a) Definition of the Turing Machine.

(b) URM computable functions are Turing computable.

(c) Undecidability of the Turing Halting Problem

# (a) Definition of TMs

- There are two problems with the model of a URM:
- Execution of a single URM instruction might take arbitrarily long:
  - Consider $\mathrm{succ}(n)$.
  - If $R_n$ contains in binary $\underbrace{111\cdots111}_{k \text{ times}}$, this instruction replaces it by $1\underbrace{000\cdots000}_{k \text{ times}}$.
  - We have to replace $k$ symbols $1$ by $0$.
  - $k$ is arbitrary
    $\rightarrow$ this single step might take arbitrarily long time.

# First Problem of URMs

- That incrementing a number by one takes arbitrarily many steps happens on a real computer as well:
  - If we want to represent arbitrary big numbers on the computer, we have to represent them by multiple machine integers
    - Then incrementing a number by one will correspond to arbitrarily many machine instructions (although usually only a few).
  - However, often in complexity theory this problem is ignored because the effect is marginal in real applications.
    - The exception are applications in which very big integers occur, e.g. tests for primality. There this effect cannot be ignored any more.

# First Problem of URMs

- If one takes this effect into account, one needs in many examples to multiply the running time by a factor of $\ln(n)$, where $n$ is the largest number occurring.
- Therefore URMs unsuitable as a basis for defining the precise complexity of algorithms.
- However, there are theorems linking complexity of URMs to actual complexities of algorithms.

# Second Problem of URMs

- We aim at a notion of computability, which covers all possible ways of computing something, independently of any concrete machine.
- URMs are a model of computation which covers current standard computers.
- However, there might be completely different notions of computability, based on symbolic manipulations of a sequence of characters, where it might be more complicated to see directly that all such computations can be simulated by a URM.
- It is more easy to see that such notions are covered by the Turing machine model of computation.

# Idea of a Turing Machine

- Steps in this formulation:
  - Algorithm should be deterministic.
    $\rightarrow$ The agent will use only finitely many symbols, put at discrete positions on the paper.

| | | 1 | 5 | . | 1 | 6 | = | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 1 | 5 | | |
| | | | | | | | | 9 | 0 | | |
| | | | | | | | | $-$ | $-$ | $-$ | |
| | | | | | | | | 2 | 4 | 0 | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |

# Idea of a Turing Machine

- Idea of a Turing machine (**TM**):
  Analysis of a computation carried out by a human being (**agent**) on a piece of paper.
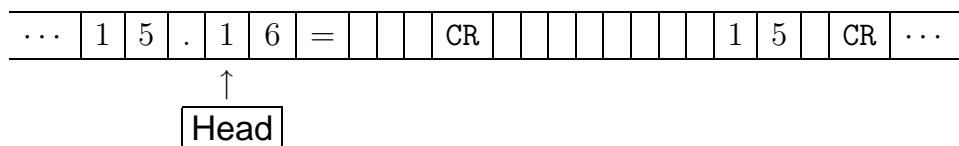
$$15 . 16 =$$
$$15$$
$$\frac{90}{240}$$

# Idea of a Turing Machine

- We can replace a two-dimensional piece of paper by one potentially infinite tape, by using a special symbol for a line break.
- Each entry on this tape is called a **cell**:

| $\cdots$ | 1 | 5 | . | 1 | 6 | = | | | | CR | | | | | | | | | 1 | 5 | | CR | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

# Steps in Formalising TMs

- In the real situation, agent can look at several cells at the same time,
  but bounded by his physical capability.
  Can be simulated by looking at one cell only at any time, and moving around in order to get information about neighbouring cells.

$$\cdots \boxed{\phantom{x}} \boxed{1} \boxed{5} \boxed{.} \boxed{1} \boxed{6} \boxed{=} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{CR} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{1} \boxed{5} \boxed{\phantom{x}} \boxed{CR} \cdots$$
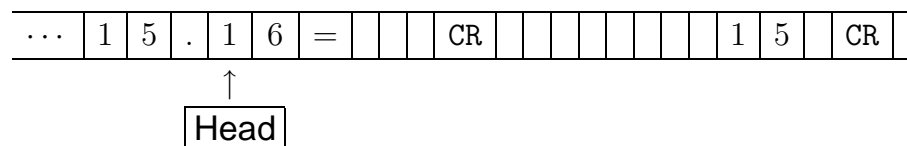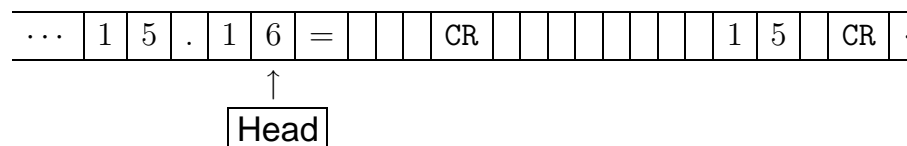
↑
Head

# Steps in Formalising TMs

- In the real situation, an agent can make arbitrary jumps, but bounded by the physical ability of the agent.
  Each such jump can be simulated by finitely many one-step jumps.
  → Restriction to one-step movements.

$$\cdots \boxed{\phantom{x}} \boxed{1} \boxed{5} \boxed{.} \boxed{1} \boxed{6} \boxed{=} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{CR} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{1} \boxed{5} \boxed{\phantom{x}} \boxed{CR}$$
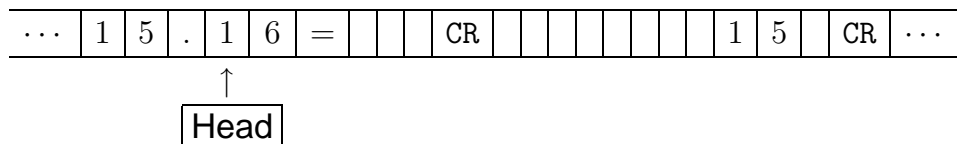
↑
Head

# Steps in Formalising TMs

- In the real situation, an agent can make arbitrary jumps, but bounded by the physical ability of the agent.
  Each such jump can be simulated by finitely many one-step jumps.
  → Restriction to one-step movements.

# Steps in Formalising TMs

- In the real situation, an agent can make arbitrary jumps, but bounded by the physical ability of the agent.
  Each such jump can be simulated by finitely many one-step jumps.
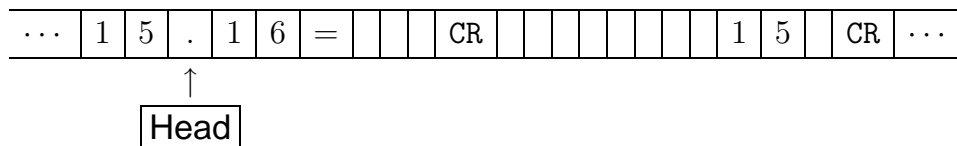  → Restriction to one-step movements.

$$\cdots \boxed{\phantom{x}} \boxed{1} \boxed{5} \boxed{.} \boxed{1} \boxed{6} \boxed{=} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{CR} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{\phantom{x}} \boxed{1} \boxed{5} \boxed{\phantom{x}} \boxed{CR}$$

↑
Head

# Steps in Formalising TMs

- In the real situation, an agent can make arbitrary jumps, but bounded by the physical ability of the agent.
  Each such jump can be simulated by finitely many one-step jumps.
  $\rightarrow$ Restriction to one-step movements.

| $\cdots$ | 1 | 5 | . | 1 | 6 | = |  |  |  | CR |  |  |  |  |  |  |  | 1 | 5 |  | CR | $\cdots$ |

$\uparrow$

Head

# Steps in Formalising TMs

- Agent operates purely mechanistically:
  Reads a symbol, and depending on it changes it and makes a movement.
  Agent himself will have only finite memory.
  $\rightarrow$ There is a finite state of the agent, and, depending on the state and the symbol at the head, a next state, a new symbol, and a movement is chosen.
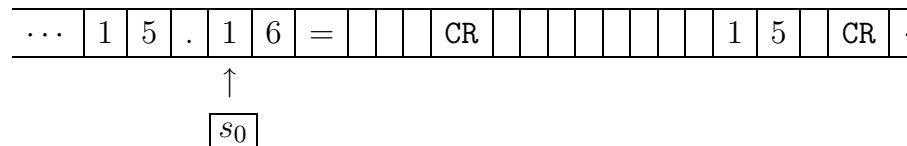
# Steps in Formalising TMs

- In the real situation, an agent can make arbitrary jumps, but bounded by the physical ability of the agent.
  Each such jump can be simulated by finitely many one-step jumps.
  $\rightarrow$ Restriction to one-step movements.

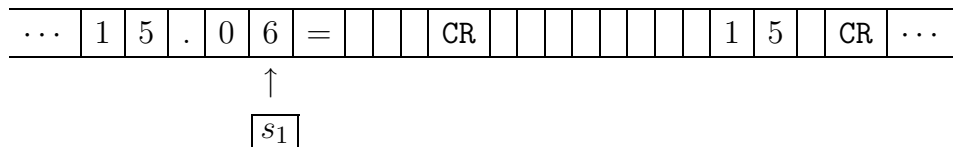| $\cdots$ | 1 | 5 | . | 1 | 6 | = |  |  |  | CR |  |  |  |  |  |  |  | 1 | 5 |  | CR | $\cdots$ |

$\uparrow$

Head

# Steps in Formalising TMs

- Agent operates purely mechanistically:
  Reads a symbol, and depending on it changes it and makes a movement.
  Agent himself will have only finite memory.
  $\rightarrow$ There is a finite state of the agent, and, depending on the state and the symbol at the head, a next state, a new symbol, and a movement is chosen.

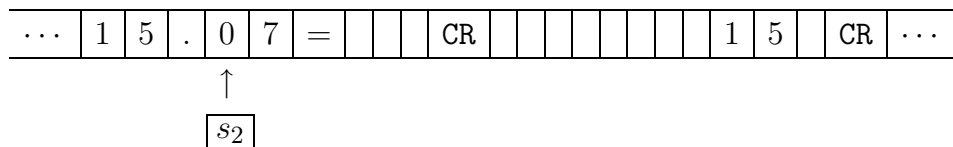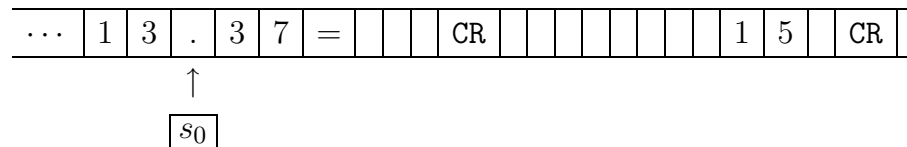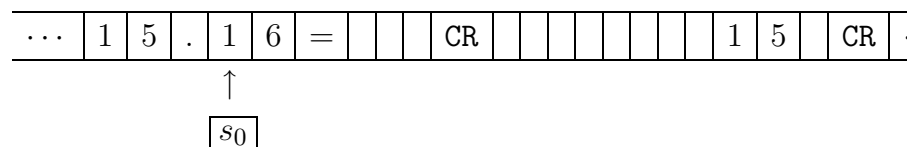| $\cdots$ | 1 | 5 | . | 1 | 6 | = |  |  |  | CR |  |  |  |  |  |  |  | 1 | 5 |  | CR |

$\uparrow$

$s_0$

# Steps in Formalising TMs

- Agent operates purely mechanistically:
  Reads a symbol, and depending on it changes it and makes a movement.
  Agent himself will have only finite memory.
  $\rightarrow$ There is a finite state of the agent, and, depending on the state and the symbol at the head, a next state, a new symbol, and a movement is chosen.

$\cdots$ | 1 | 5 | . | 0 | 6 | = | | | | CR | | | | | | | | | 1 | 5 | | CR | $\cdots$

$\uparrow$

$\boxed{s_1}$

# Steps in Formalising TMs

$\cdots$ | 1 | 5 | . | 0 | 7 | = | | | | CR | | | | | | | | | 1 | 5 | | CR | $\cdots$

$\uparrow$

$\boxed{s_2}$

# Steps in Formalising TMs

$\cdots$ | 1 | 3 | . | 3 | 7 | = | | | | CR | | | | | | | | | 1 | 5 | | CR |

$\uparrow$

$\boxed{s_0}$

# Definition of TMs

$\cdots$ | 1 | 5 | . | 1 | 6 | = | | | | CR | | | | | | | | | 1 | 5 | | CR |

$\uparrow$

$\boxed{s_0}$

- A **Turing machine** is a five tuple (or quintuple) $(\Sigma, \mathsf{S}, \mathsf{I}, {\llcorner\lrcorner}, s_0)$, where
  - $\Sigma$ is a finite set of symbols, called the **alphabet** of the Turing machine. On the tape, the symbols in $\Sigma$ will be written.
  - S is a finite set of states.

# Definition of TMs

| | 1 | 5 | . | 1 | 6 | = | | | | CR | | | | | | | | 1 | 5 | | CR | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\cdots$ | | | | | | | | | | | | | | | | | | | | | | $\cdots$ |

$$\uparrow$$
$$\boxed{s_0}$$
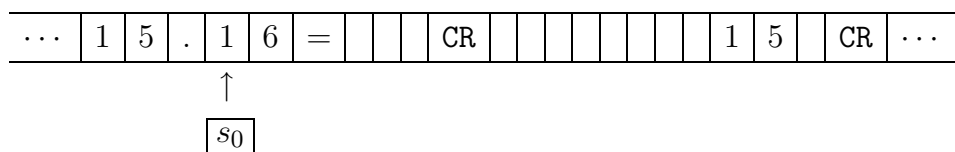
- I is a finite sets of quintuples $(s, a, s', a', D)$, where
  - $s, s' \in \mathsf{S}$,
  - $a, a' \in \Sigma$
  - $D \in \{\mathsf{L}, \mathsf{R}\}$,
  
  s.t. for every $s \in \mathsf{S}$, $a \in \Sigma$ there is **at most one** $s', a', D$ s.t. $(s, a, s', a', D) \in \mathsf{I}$.
  The elements of I are called **instructions**.

- $\llcorner\lrcorner \in \Sigma$ (a symbol for blank).

- $s_0 \in \mathsf{S}$ (the initial state).

# Meaning of Instructions

- A instruction $(s, a, s', a', D) \in \mathsf{I}$ means the following:

  - If the Turing machine is in state $s$, and the symbol at position of the head is $a$, then
    - the state is changed to $s'$,
    - the symbol at this position is changed to $a'$,
    - if $D = \mathsf{L}$, the head moves left,
    - if $D = \mathsf{R}$, the head moves right.

**Example:**
$(s_0, 1, s_1, 0, \mathsf{R})$
$(s_1, 6, s_2, 7, \mathsf{L})$

# Meaning of Instructions

- A instruction $(s, a, s', a', D) \in \mathsf{I}$ means the following:

  - If the Turing machine is in state $s$, and the symbol at position of the head is $a$, then
    - the state is changed to $s'$,
    - the symbol at this position is changed to $a'$,
    - if $D = \mathsf{L}$, the head moves left,
    - if $D = \mathsf{R}$, the head moves right.

# Meaning of Instructions

- A instruction $(s, a, s', a', D) \in \mathsf{I}$ means the following:

  - If the Turing machine is in state $s$, and the symbol at position of the head is $a$, then
    - the state is changed to $s'$,
    - the symbol at this position is changed to $a'$,
    - if $D = \mathsf{L}$, the head moves left,
    - if $D = \mathsf{R}$, the head moves right.

**Example:**
$(s_0, 1, s_1, 0, \mathsf{R})$
$(s_1, 6, s_2, 7, \mathsf{L})$

| | 1 | 5 | . | 1 | 6 | = | | | | CR | | | | | | | | 1 | 5 | | CR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\cdots$ | | | | | | | | | | | | | | | | | | | | | |

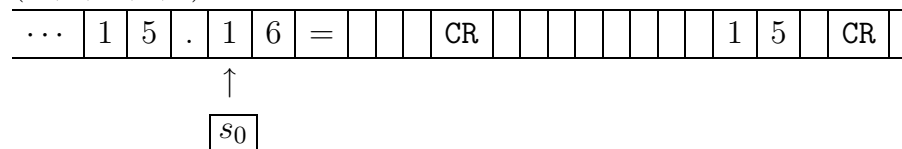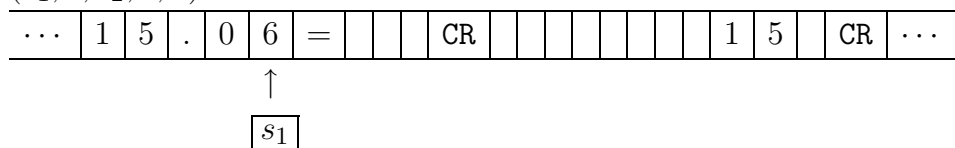$$\uparrow$$
$$\boxed{s_0}$$

# Meaning of Instructions

- A instruction $(s, a, s', a', D) \in$ I means the following:
  - If the Turing machine is in state $s$, and the symbol at position of the head is $a$, then
    - the state is changed to $s'$,
    - the symbol at this position is changed to $a'$,
    - if $D = $ L, the head moves left,
    - if $D = $ R, the head moves right.

**Example:**
$(s_0, 1, s_1, 0, \mathsf{R})$
$(s_1, 6, s_2, 7, \mathsf{L})$

| $\cdots$ | 1 | 5 | . | 0 | 6 | = | | | | CR | | | | | | | | 1 | 5 | | CR | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

$\uparrow$

$\boxed{s_1}$

# Meaning of Instructions

- Note that for the above it is important that for every $s \in$ S, $a \in \Sigma$ there is **at most one** $s', a', D$ s.t. $(s, a, s', a', D) \in$ S.
  - Without this condition, there might be more than one choice of selecting a new tape symbol, next state and direction.
  - If we omit this definition, we obtain a **non-deterministic TM**. In this case the machine selects in each step one of the possible choices (provided there exist one) at random.

- If the Turing machine is in a state $s$ and reads symbol $a$ at his head, and there are no $s', a', D$ s.t. $(s, a, s', a', D) \in$ I, then the Turing machine stops.
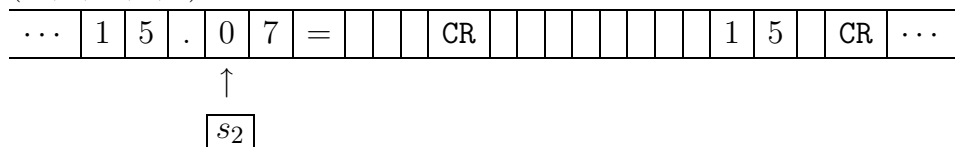
# Meaning of Instructions

- A instruction $(s, a, s', a', D) \in$ I means the following:
  - If the Turing machine is in state $s$, and the symbol at position of the head is $a$, then
    - the state is changed to $s'$,
    - the symbol at this position is changed to $a'$,
    - if $D = $ L, the head moves left,
    - if $D = $ R, the head moves right.

**Example:**
$(s_0, 1, s_1, 0, \mathsf{R})$
$(s_1, 6, s_2, 7, \mathsf{L})$

| $\cdots$ | 1 | 5 | . | 0 | 7 | = | | | | CR | | | | | | | | 1 | 5 | | CR | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

$\uparrow$

$\boxed{s_2}$

# TM Architecture vs. TM Program

- As for URMs a TM means both the TM architecture and the TM program.
  - The TM architecture describes that a TM has a tape, a head, a state, and how it is executed.
  - The TM program consists of the alphabet on the tape, the set of states, the instructions, the symbol for blank and the initial state.
- When asked to define a TM which has a certain behaviour oen usually actually asks for a TM program, such that a TM with this program has this behaviour.
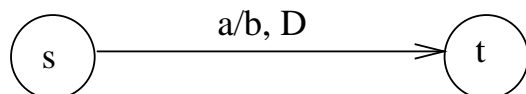
# Visualisation of TMs

- A TM

$$(\Sigma, \mathsf{S}, \mathsf{I}, {}_{\llcorner\lrcorner}, s_0)$$

can be visualised by a labelled graph as follows:
  - Vertices: states (i.e. $\mathsf{S}$).
  - Edges: If $(s, a, t, b, D) \in \mathsf{I}$, then there is an edge



- Furthermore we write an arrow to the initial state coming from nowhere.
- If there are several vertices from $s$ to $s'$, one draws only one arrow with one label for each vertex.

# Example

The Turing machine with initial state $s_0$ and instructions

$$\begin{aligned}\{&(s_0, 0, s_0, 0, \mathsf{R}),\\ &(s_0, 1, s_0, 0, \mathsf{R}),\\ &(s_0, {}_{\llcorner\lrcorner}, s_1, {}_{\llcorner\lrcorner}, \mathsf{L}),\\ &(s_1, 0, s_1, 0, \mathsf{L}),\\ &(s_1, {}_{\llcorner\lrcorner}, s_2, {}_{\llcorner\lrcorner}, \mathsf{R})\}\end{aligned}$$

is visualised as follows (we write $B$ instead of ${}_{\llcorner\lrcorner}$):
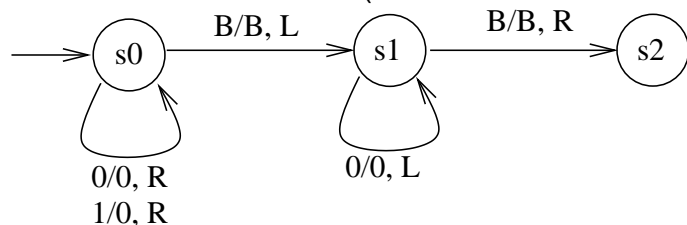
# Example

- The TM on the previous slide sets the binary number the head is pointing to to zero, provided to the left of the head there are is a blank.
- **Exercise**:
  - This example assumes that the TM points to the left most digit of a binary number.
  - Modify this TM, so that it works as well if the TM points initially to any digit of a binary number.

# Example of a TM

- Development of a TM with $\Sigma = \{0, 1, {}_{\llcorner\lrcorner}\}$,
  - where ${}_{\llcorner\lrcorner}$ is the symbol for the blank entry.
- Functionality of the TM:
  - Assume initially the following:
    - The tape contains binary number,
    - The rest of the tape contains ${}_{\llcorner\lrcorner}$.
    - The head points to any digit of the number.
    - The TM in state $s_0$.
  - Then the TM stops after finitely many steps and then
    - the tape contains the original number incremented by one,
    - the rest of tape contains ${}_{\llcorner\lrcorner}$,
    - the head points to most significant bit.
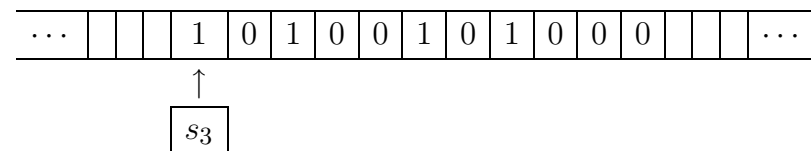
# Equivalent Representations

- The pictorial representation is equivalent to the set of instructions plus an initial state.

- Therefore a TM can both be given by listing its instructions and by the pictorial representation.

- Furthermore the only relevant sets of instructions are those occurring in the picturial representation. Similarly for the set of symbols on the tape.

- Therefore, assuming that the blank symbol is canonical, we can take the picturial representation as the complete definition of a TM (with states being the set of states occurring in the diagram, and alphabet consisting of the canonical blank symbol and the states occurring in the diagram).

# Example

Initially

| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |

$\uparrow$

$\boxed{s_0}$

# Example

Initially

| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |

$\uparrow$

$\boxed{s_0}$

Finally

| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | | | $\cdots$ |

$\uparrow$

$\boxed{s_3}$
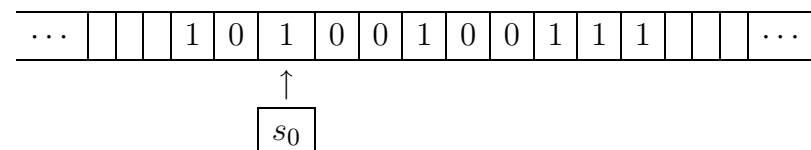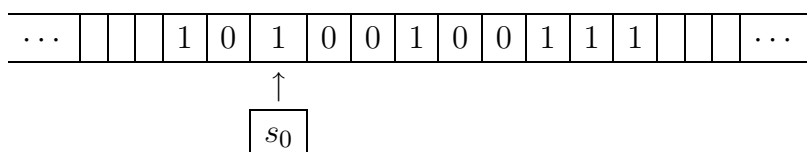
# Construction of the TM

- TM is $(\{0, 1, \llcorner\lrcorner\}, S, I, \llcorner\lrcorner, s_0)$.

- States S and instructions I developed in the following.

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\llcorner\lrcorner$, move head left, leave symbol again as it is.
- Achieved by the following instructions:

$(s_0, 0, s_0, 0, \mathsf{R})$
$(s_0, 1, s_0, 1, \mathsf{R})$
$(s_0, \llcorner\lrcorner, s_1, \llcorner\lrcorner, \mathsf{L})$

s0 —B/B, L→ s1
0/0, R
1/1, R

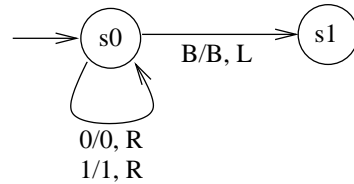- At the end TM is in state $s_1$.

4-22

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\llcorner\lrcorner$, move head left, leave symbol again as it is.
- Achieved by the following instructions:

$(s_0, 0, s_0, 0, \mathsf{R})$
$(s_0, 1, s_0, 1, \mathsf{R})$
$(s_0, \llcorner\lrcorner, s_1, \llcorner\lrcorner, \mathsf{L})$

s0 —B/B, L→ s1
0/0, R
1/1, R

- At the end TM is in state $s_1$.

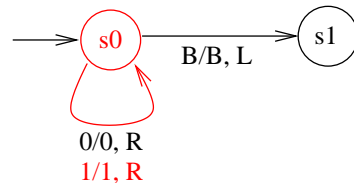| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |

↑
$s_0$

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\llcorner\lrcorner$, move head left, leave symbol again as it is.
- Achieved by the following instructions:

$(s_0, 0, s_0, 0, \mathsf{R})$
$(s_0, 1, s_0, 1, \mathsf{R})$
$(s_0, \llcorner\lrcorner, s_1, \llcorner\lrcorner, \mathsf{L})$

s0 —B/B, L→ s1
0/0, R
1/1, R

- At the end TM is in state $s_1$.

| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |

↑
$s_0$

4-22

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\llcorner\lrcorner$, move head left, leave symbol again as it is.
- Achieved by the following instructions:

$(s_0, 0, s_0, 0, \mathsf{R})$
$(s_0, 1, s_0, 1, \mathsf{R})$
$(s_0, \llcorner\lrcorner, s_1, \llcorner\lrcorner, \mathsf{L})$

s0 —B/B, L→ s1
0/0, R
1/1, R

- At the end TM is in state $s_1$.

| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |

↑
$s_0$

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\sqcup$, move head left, leave symbol again as it is.
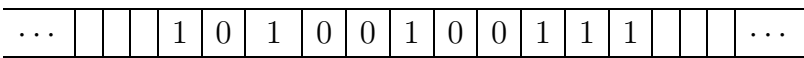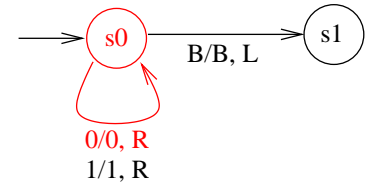- Achieved by the following instructions:

$(s_0, 0, s_0, 0, \mathsf{R})$
$\color{red}(s_0, 1, s_0, 1, \mathsf{R})$
$(s_0, \sqcup, s_1, \sqcup, \mathsf{L})$



B/B, L

0/0, R
1/1, R

- At the end TM is in state $s_1$.



| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |

↑

$s_0$

---

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\sqcup$, move head left, leave symbol again as it is.
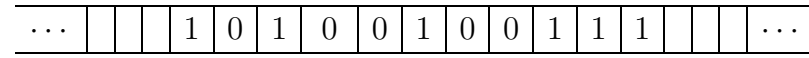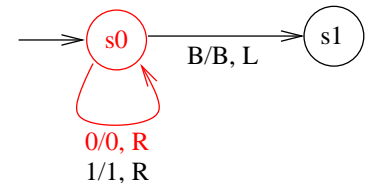- Achieved by the following instructions:

$\color{red}(s_0, 0, s_0, 0, \mathsf{R})$
$(s_0, 1, s_0, 1, \mathsf{R})$
$(s_0, \sqcup, s_1, \sqcup, \mathsf{L})$



B/B, L

0/0, R
1/1, R

- At the end TM is in state $s_1$.



| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |

↑

$s_0$

---

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\sqcup$, move head left, leave symbol again as it is.
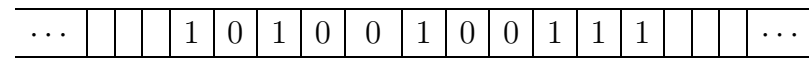- Achieved by the following instructions:

$(s_0, 0, s_0, 0, \mathsf{R})$
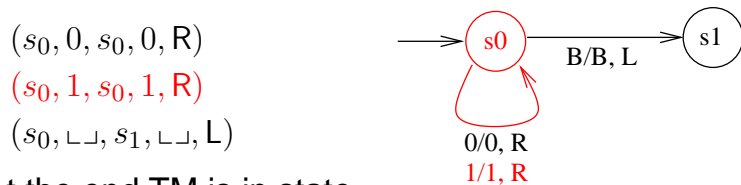$(s_0, 1, s_0, 1, \mathsf{R})$
$(s_0, \sqcup, s_1, \sqcup, \mathsf{L})$



B/B, L

0/0, R
1/1, R

- At the end TM is in state $s_1$.

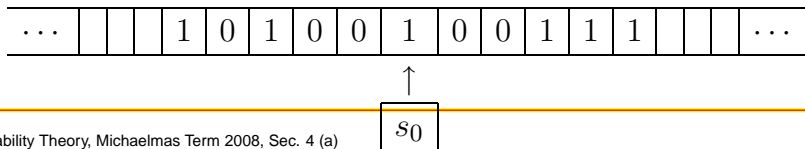| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |

↑

$s_0$

---

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\sqcup$, move head left, leave symbol again as it is.
- Achieved by the following instructions:

$(s_0, 0, s_0, 0, \mathsf{R})$
$\color{red}(s_0, 1, s_0, 1, \mathsf{R})$
$(s_0, \sqcup, s_1, \sqcup, \mathsf{L})$



B/B, L

0/0, R
1/1, R

- At the end TM is in state $s_1$.

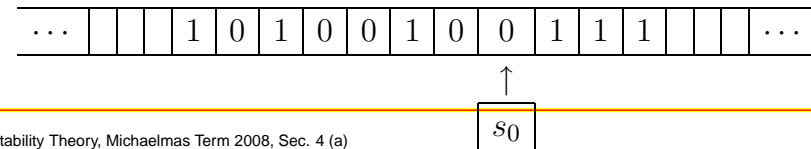| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |

↑

$s_0$

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\llcorner\lrcorner$, move head left, leave symbol again as it is.
- Achieved by the following instructions:

$(s_0, 0, s_0, 0, \mathsf{R})$

$(s_0, 1, s_0, 1, \mathsf{R})$

$(s_0, \llcorner\lrcorner, s_1, \llcorner\lrcorner, \mathsf{L})$

s0 — B/B, L → s1

0/0, R
1/1, R

- At the end TM is in state $s_1$.

| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

↑

$s_0$

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\llcorner\lrcorner$, move head left, leave symbol again as it is.
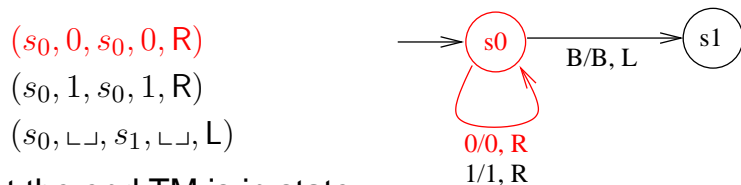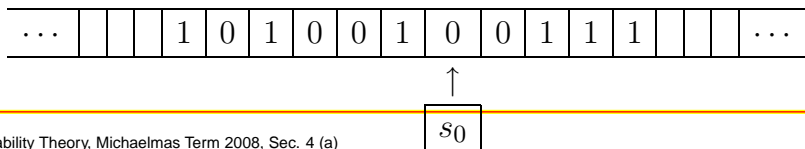- Achieved by the following instructions:

$(s_0, 0, s_0, 0, \mathsf{R})$

$(s_0, 1, s_0, 1, \mathsf{R})$

$(s_0, \llcorner\lrcorner, s_1, \llcorner\lrcorner, \mathsf{L})$

s0 — B/B, L → s1

0/0, R
1/1, R

- At the end TM is in state $s_1$.

| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

↑

$s_0$

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\llcorner\lrcorner$, move head left, leave symbol again as it is.
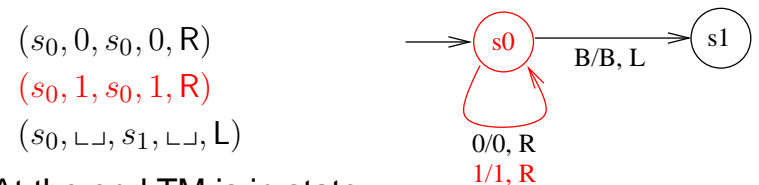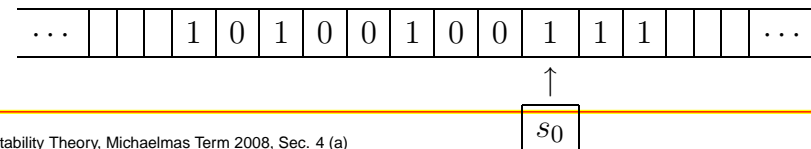- Achieved by the following instructions:

$(s_0, 0, s_0, 0, \mathsf{R})$

$(s_0, 1, s_0, 1, \mathsf{R})$

$(s_0, \llcorner\lrcorner, s_1, \llcorner\lrcorner, \mathsf{L})$

s0 — B/B, L → s1

0/0, R
1/1, R

- At the end TM is in state $s_1$.

| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

↑

$s_0$

# Step 1

- Initially, move head to least significant bit.
  - I.e. as long as symbol at head is $0$ or $1$, move right, leave symbol as it is.
  - If symbol is $\llcorner\lrcorner$, move head left, leave symbol again as it is.
- Achieved by the following instructions:

$(s_0, 0, s_0, 0, \mathsf{R})$

$(s_0, 1, s_0, 1, \mathsf{R})$

$(s_0, \llcorner\lrcorner, s_1, \llcorner\lrcorner, \mathsf{L})$

s0 — B/B, L → s1

0/0, R
1/1, R

- At the end TM is in state $s_1$.

| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

↑

$s_1$

# Step 2

Increasing a binary number $b$ done as follows:

- **Case number consists of** $1$ **only:**
    - I.e. $b = (\underbrace{111\cdots111}_{k \text{ times}})_2$.
    - $b+1 = (1\underbrace{000\cdots000}_{k \text{ times}})_2$.
    - Obtained by replacing all ones by zeros and then replacing the first blank symbol by $1$.
    - That's what happens when we add by hand:

$$
\begin{array}{ccccccccc}
 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
+ & & & & & & & 1 \\
\hline
\end{array}
$$

# Step 2

Increasing a binary number $b$ done as follows:

- **Case number consists of** $1$ **only:**
    - I.e. $b = (\underbrace{111\cdots111}_{k \text{ times}})_2$.
    - $b+1 = (1\underbrace{000\cdots000}_{k \text{ times}})_2$.
    - Obtained by replacing all ones by zeros and then replacing the first blank symbol by $1$.
    - That's what happens when we add by hand:

$$
\begin{array}{ccccccccc}
 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
+ & & & & & & & 1 \\
\hline
 & & & & & & 1 & \\
\hline
 & & & & & & & 0 \\
\end{array}
$$

# Step 2

Increasing a binary number $b$ done as follows:

- **Case number consists of** $1$ **only:**
    - I.e. $b = (\underbrace{111\cdots111}_{k \text{ times}})_2$.
    - $b+1 = (1\underbrace{000\cdots000}_{k \text{ times}})_2$.
    - Obtained by replacing all ones by zeros and then replacing the first blank symbol by $1$.
    - That's what happens when we add by hand:

$$
\begin{array}{ccccccccc}
 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
+ & & & & & & & 1 \\
\hline
 & & & & & & 1 & 1 \\
\hline
 & & & & & & 0 & 0 \\
\end{array}
$$

# Step 2

Increasing a binary number $b$ done as follows:

- **Case number consists of** $1$ **only:**
    - I.e. $b = (\underbrace{111\cdots111}_{k \text{ times}})_2$.
    - $b+1 = (1\underbrace{000\cdots000}_{k \text{ times}})_2$.
    - Obtained by replacing all ones by zeros and then replacing the first blank symbol by $1$.
    - That's what happens when we add by hand:

$$
\begin{array}{ccccccccc}
 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
+ & & & & & & & 1 \\
\hline
 & & & & & 1 & 1 & 1 \\
\hline
 & & & & & 0 & 0 & 0 \\
\end{array}
$$

# Step 2

Increasing a binary number $b$ done as follows:

- **Case number consists of** $1$ **only:**
  - I.e. $b = (\underbrace{111\cdots111}_{k\text{ times}})_2$.
  - $b + 1 = (1\underbrace{000\cdots000}_{k\text{ times}})_2$.
  - Obtained by replacing all ones by zeros and then replacing the first blank symbol by $1$.
  - That's what happens when we add by hand:

$$
\begin{array}{ccccccc}
  & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
+ &   &   &   &   &   &   & 1 \\
\hline
  &   &   &   & 1 & 1 & 1 & 1 \\
\hline
  &   &   &   & 0 & 0 & 0 & 0
\end{array}
$$

# Step 2

Increasing a binary number $b$ done as follows:

- **Case number consists of** $1$ **only:**
  - I.e. $b = (\underbrace{111\cdots111}_{k\text{ times}})_2$.
  - $b + 1 = (1\underbrace{000\cdots000}_{k\text{ times}})_2$.
  - Obtained by replacing all ones by zeros and then replacing the first blank symbol by $1$.
  - That's what happens when we add by hand:

$$
\begin{array}{ccccccc}
  & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
+ &   &   &   &   &   &   & 1 \\
\hline
  &   &   &   & 1 & 1 & 1 & 1 \\
\hline
  &   &   & 1 & 0 & 0 & 0 & 0
\end{array}
$$

# Step 2

Increasing a binary number $b$ done as follows:

- **Case number consists of** $1$ **only:**
  - I.e. $b = (\underbrace{111\cdots111}_{k\text{ times}})_2$.
  - $b + 1 = (1\underbrace{000\cdots000}_{k\text{ times}})_2$.
  - Obtained by replacing all ones by zeros and then replacing the first blank symbol by $1$.
  - That's what happens when we add by hand:

$$
\begin{array}{ccccccc}
  & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
+ &   &   &   &   &   &   & 1 \\
\hline
  &   &   &   & 1 & 1 & 1 & 1 \\
\hline
  &   & 0 & 1 & 0 & 0 & 0 & 0
\end{array}
$$

# Step 2

Increasing a binary number $b$ done as follows:

- **Case number consists of** $1$ **only:**
  - I.e. $b = (\underbrace{111\cdots111}_{k\text{ times}})_2$.
  - $b + 1 = (1\underbrace{000\cdots000}_{k\text{ times}})_2$.
  - Obtained by replacing all ones by zeros and then replacing the first blank symbol by $1$.
  - That's what happens when we add by hand:

$$
\begin{array}{ccccccc}
  & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
+ &   &   &   &   &   &   & 1 \\
\hline
  &   &   &   & 1 & 1 & 1 & 1 \\
\hline
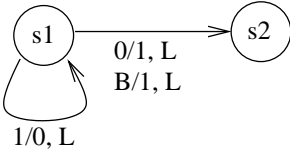  & 1 & 0 & 1 & 0 & 0 & 0 & 0
\end{array}
$$

# Step 2

- **Otherwise**:
  - Then the representation of the number contains at the end one $0$ followed by ones only.
    Includes case where the least significant digit is $0$.
    - Example 1: $b = (010001\mathbf{0111})_2$, one $0$ followed by $3$ ones.
    - Example 2: $b = (010001001\mathbf{0})_2$, least significant digit is $0$.
  - Let $b = (b_0 b_1 \cdots b_k 0 \underbrace{111 \cdots 111}_{l \text{ times}})_2$.
  - $b + 1$ obtained by replacing the final block of ones by $0$ and the $0$ by $1$:
    $b + 1 = (b_0 b_1 \cdots b_k 1 \underbrace{000 \cdots 000}_{l \text{ times}})_2$.
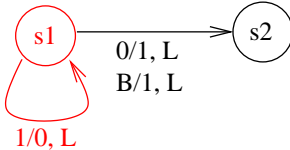
# Step 2 – General Situation

- We have to replace, as long as we find ones, the ones by zeros, and move left, until we encounter a $0$ or a $\llcorner\lrcorner$, which is replaced by a $1$.

- So we need a new state $s_2$, and the following instructions
  $(s_1, 1, s_1, 0, \mathsf{L})$
  $(s_1, 0, s_2, 1, \mathsf{L})$
  $(s_1, \llcorner\lrcorner, s_2, 1, \mathsf{L})$

  s1 ──0/1, L── s2
      B/1, L
  1/0, L

- At the end the head will be one field to the left of the $1$ written, and the state will be $s_2$.
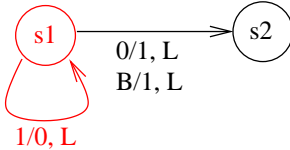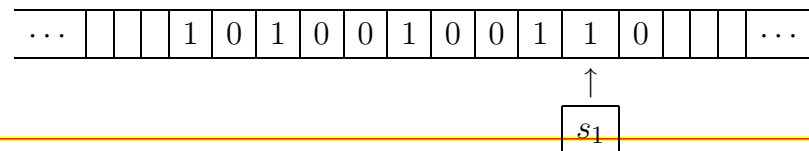
# Step 2 – General Situation

- We have to replace, as long as we find ones, the ones by zeros, and move left, until we encounter a $0$ or a $\llcorner\lrcorner$, which is replaced by a $1$.

- So we need a new state $s_2$, and the following instructions
  $(s_1, 1, s_1, 0, \mathsf{L})$
  $(s_1, 0, s_2, 1, \mathsf{L})$
  $(s_1, \llcorner\lrcorner, s_2, 1, \mathsf{L})$

  s1 ──0/1, L── s2
      B/1, L
  1/0, L

- At the end the head will be one field to the left of the $1$ written, and the state will be $s_2$.

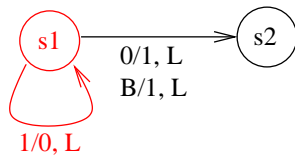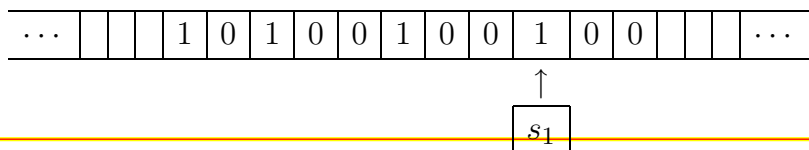| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

$\uparrow$

$s_1$

# Step 2 – General Situation

- We have to replace, as long as we find ones, the ones by zeros, and move left, until we encounter a $0$ or a $\llcorner\lrcorner$, which is replaced by a $1$.

- So we need a new state $s_2$, and the following instructions
  $(s_1, 1, s_1, 0, \mathsf{L})$
  $(s_1, 0, s_2, 1, \mathsf{L})$
  $(s_1, \llcorner\lrcorner, s_2, 1, \mathsf{L})$

  s1 ──0/1, L── s2
      B/1, L
  1/0, L

- At the end the head will be one field to the left of the $1$ written, and the state will be $s_2$.

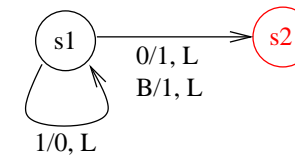| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | | | | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

$\uparrow$
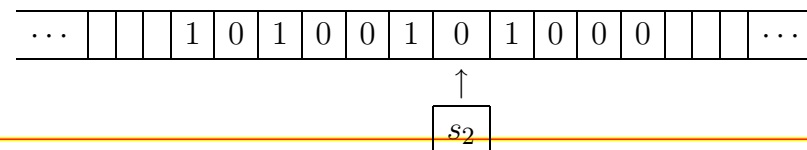
$s_1$

# Step 2 – General Situation

- We have to replace, as long as we find ones, the ones by zeros, and move left, until we encounter a $0$ or a $\llcorner\lrcorner$, which is replaced by a $1$.

- So we need a new state $s_2$, and the following instructions

$$(s_1, 1, s_1, 0, \mathsf{L})$$
$$(s_1, 0, s_2, 1, \mathsf{L})$$
$$(s_1, \llcorner\lrcorner, s_2, 1, \mathsf{L})$$

- At the end the head will be one field to the left of the $1$ written, and the state will be $s_2$.

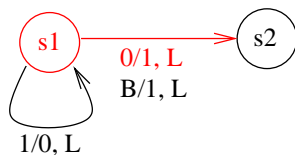| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | | | | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

$\uparrow$

$\boxed{s_1}$

# Step 2 – General Situation

- We have to replace, as long as we find ones, the ones by zeros, and move left, until we encounter a $0$ or a $\llcorner\lrcorner$, which is replaced by a $1$.

- So we need a new state $s_2$, and the following instructions

$$(s_1, 1, s_1, 0, \mathsf{L})$$
$$(s_1, 0, s_2, 1, \mathsf{L})$$
$$(s_1, \llcorner\lrcorner, s_2, 1, \mathsf{L})$$

- At the end the head will be one field to the left of the $1$ written, and the state will be $s_2$.

| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | | | | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

$\uparrow$

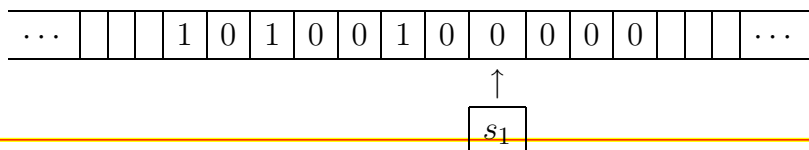$\boxed{s_1}$

# Step 2 – General Situation

- We have to replace, as long as we find ones, the ones by zeros, and move left, until we encounter a $0$ or a $\llcorner\lrcorner$, which is replaced by a $1$.

- So we need a new state $s_2$, and the following instructions

$$(s_1, 1, s_1, 0, \mathsf{L})$$
$$(s_1, 0, s_2, 1, \mathsf{L})$$
$$(s_1, \llcorner\lrcorner, s_2, 1, \mathsf{L})$$

- At the end the head will be one field to the left of the $1$ written, and the state will be $s_2$.

| $\cdots$ | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | | | | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

$\uparrow$

$\boxed{s_2}$

# Step 3

Finally, we have to move the most significant bit, which is done as follows

$$(s_2, 0, s_2, 0, \mathsf{L})$$
$$(s_2, 1, s_2, 1, \mathsf{L})$$
$$(s_2, \llcorner\lrcorner, s_3, \llcorner\lrcorner, \mathsf{R})$$

The program terminates in state $s_3$.

# Step 3

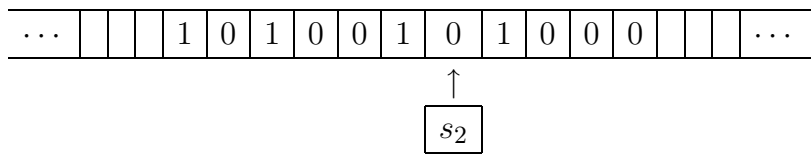Finally, we have to move the most significant bit, which is done as follows

$(s_2, 0, s_2, 0, \mathsf{L})$
$(s_2, 1, s_2, 1, \mathsf{L})$
$(s_2, \llcorner \lrcorner, s_3, \llcorner \lrcorner, \mathsf{R})$



The program terminates in state $s_3$.

# Step 3

Finally, we have to move the most significant bit, which is done as follows

$(s_2, 0, s_2, 0, \mathsf{L})$
$(s_2, 1, s_2, 1, \mathsf{L})$
$(s_2, \llcorner \lrcorner, s_3, \llcorner \lrcorner, \mathsf{R})$



The program terminates in state $s_3$.

# Step 3

Finally, we have to move the most significant bit, which is done as follows

$(s_2, 0, s_2, 0, \mathsf{L})$
$(s_2, 1, s_2, 1, \mathsf{L})$
$(s_2, \llcorner \lrcorner, s_3, \llcorner \lrcorner, \mathsf{R})$



The program terminates in state $s_3$.

# Step 3

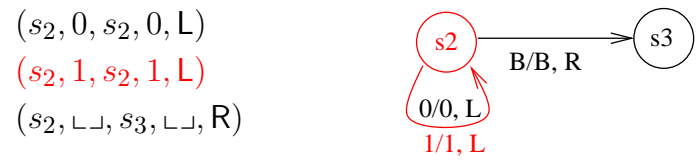Finally, we have to move the most significant bit, which is done as follows

$(s_2, 0, s_2, 0, \mathsf{L})$
$(s_2, 1, s_2, 1, \mathsf{L})$
$(s_2, \llcorner \lrcorner, s_3, \llcorner \lrcorner, \mathsf{R})$



The program terminates in state $s_3$.

# Step 3

Finally, we have to move the most significant bit, which is done as follows

$(s_2, 0, s_2, 0, \mathsf{L})$

$(s_2, 1, s_2, 1, \mathsf{L})$

$(s_2, \llcorner\lrcorner, s_3, \llcorner\lrcorner, \mathsf{R})$



The program terminates in state $s_3$.

# Step 3

Finally, we have to move the most significant bit, which is done as follows

$(s_2, 0, s_2, 0, \mathsf{L})$

$(s_2, 1, s_2, 1, \mathsf{L})$

$(s_2, \llcorner\lrcorner, s_3, \llcorner\lrcorner, \mathsf{R})$



The program terminates in state $s_3$.

# Step 3

Finally, we have to move the most significant bit, which is done as follows

$(s_2, 0, s_2, 0, \mathsf{L})$

$(s_2, 1, s_2, 1, \mathsf{L})$

$(s_2, \llcorner\lrcorner, s_3, \llcorner\lrcorner, \mathsf{R})$



The program terminates in state $s_3$.

# Step 3

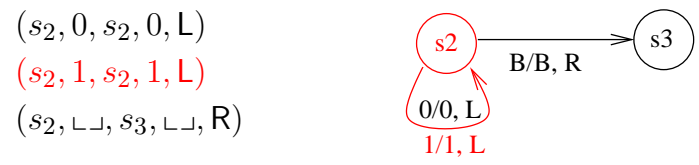Finally, we have to move the most significant bit, which is done as follows

$(s_2, 0, s_2, 0, \mathsf{L})$

$(s_2, 1, s_2, 1, \mathsf{L})$

$(s_2, \llcorner\lrcorner, s_3, \llcorner\lrcorner, \mathsf{R})$



The program terminates in state $s_3$.

# Step 3

Finally, we have to move the most significant bit, which is done as follows

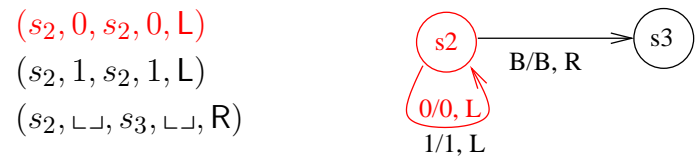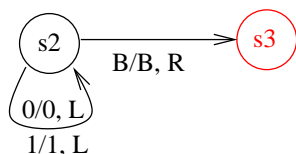$$(s_2, 0, s_2, 0, \mathsf{L})$$
$$(s_2, 1, s_2, 1, \mathsf{L})$$
$$(s_2, \llcorner\lrcorner, s_3, \llcorner\lrcorner, \mathsf{R})$$



The program terminates in state $s_3$.

# Complete TM

The complete TM is as follows:

$$(\{0, 1, \llcorner\lrcorner\},$$
$$\{s_0, s_1, s_2, s_3\},$$
$$\{(s_0, 0, s_0, 0, \mathsf{R}),$$
$$(s_0, 1, s_0, 1, \mathsf{R}),$$
$$(s_0, \llcorner\lrcorner, s_1, \llcorner\lrcorner, \mathsf{L}),$$
$$(s_1, 1, s_1, 0, \mathsf{L}),$$
$$(s_1, 0, s_2, 1, \mathsf{L}),$$
$$(s_1, \llcorner\lrcorner, s_2, 1, \mathsf{L}),$$
$$(s_2, 0, s_2, 0, \mathsf{L}),$$
$$(s_2, 1, s_2, 1, \mathsf{L}),$$
$$(s_2, \llcorner\lrcorner, s_3, \llcorner\lrcorner, \mathsf{R})\}, \llcorner\lrcorner, s_0)$$

# Complete TM
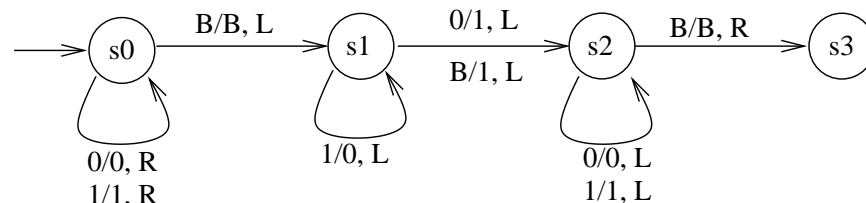
# Notation: bin

- TMs usually operate on binary numbers.
- Therefore we define for a natural number $\mathrm{bin}(n)$ as the

  sequence of digits representing the unique standard binary representation of $n$.
  - So $\mathrm{bin}(n)$ has no leading zeros, except for $\mathrm{bin}(0) := "0"$.
- Examples:
  - $\mathrm{bin}(0) = "0"$,
  - $\mathrm{bin}(1) = "1"$,
  - $\mathrm{bin}(2) = "10"$,
  - $\mathrm{bin}(3) = "11"$,
  - $\mathrm{bin}(4) = "100"$, etc.

# Notation: $\widetilde{\text{bin}}$

- In order to read off the final result, we need to interpret an arbitrary finite sequence of $0$, $1$ as a binary number, even if it has leading zeros.
- We define $\widetilde{\text{bin}}(n)$ as one of the possible binary representations of $n$, allowing leading $0$.
  - So $\widetilde{\text{bin}}(1)$ can be "$1$", "$01$", "$001$", etc.
  - In the special case $0$ we treat the empty string as one of the possible representations, so $\widetilde{\text{bin}}(0)$ can be "", "$0$", "$00$", "$000$", etc.

# $(b_0, \ldots, b_{k-1})_2$, $(d_0, \ldots, d_{k-1})_{10}$

- We write as well
  - $(b_0, \ldots, b_{k-1})_2$ for the natural number having binary representation $b_0, \ldots, b_{k-1}$), e.g.

  $$(01010)_2 = 12$$

  - $(d_0, \ldots, d_{k-1})_{10}$ for the natural number with decimal representation $d_0, \ldots, d_{k-1}$ e.g.

  $$(101)_{10} = 101$$

# Notation: $\widetilde{\text{bin}}$

- When carrying out intermediate calculations, it is easier to refer to $\widetilde{\text{bin}}(n)$ rather than $\text{bin}(n)$
  - E.g. we can set a number on the tape easily to an element of $\widetilde{\text{bin}}(0)$ by overwriting it with $0$s.
  - In order to set it to $\text{bin}(0)$ one would need to make sure that exactly one $0$ remains. Then one usually has to shift left the content of the tape to the right of the original number.

# Definition 4.1

Let $T = (\Sigma, S, I, \llcorner\lrcorner, s_0)$ be a Turing machine with $\{0, 1\} \subseteq \Sigma$.
Define for every $k \in \mathbb{N}$ $T^{(k)} : \mathbb{N}^k \xrightarrow{\sim} \mathbb{N}$, where

$T^{(k)}(a_0, \ldots, a_{k-1})$ is computed as follows:

- **Initialisation:**
  - We write on the tape
    $\text{bin}(a_0)\llcorner\lrcorner\text{bin}(a_1)\llcorner\lrcorner\cdots\llcorner\lrcorner\text{bin}(a_{k-1})$.
    - E.g. if $k = 3$, $a_0 = 0$, $a_1 = 3$, $a_2 = 2$ then we write $0\llcorner\lrcorner11\llcorner\lrcorner10$.
  - All other cells contain $\llcorner\lrcorner$.
  - The head is at the left most bit of the arguments written on the tape.
  - The state is set to $s_0$.
- **Iteration:** Run the TM, until it stops.

# Definition 4.1

- **Output:**
  - **Case 1:** The TM stops.
    Only finitely many cells are non-blank.
    Let tape, starting from the head-position, contain
    $b_0 b_1 \cdots b_{k-1} c$ where $b_i \in \{0, 1\}$ and $c \notin \{0, 1\}$.
    ($k$ might be $0$).
    Let
    $$a = (b_0, \ldots, b_{k-1})_2 \ .$$
    (in case $k = 0$, $a = 0$).
    This means "$b_0 \cdots b_{k-1}$" is one of the choices for
    $\widetilde{\mathrm{bin}}(a)$.
    Then
    $$\mathrm{T}^{(k)}(a_0, \ldots, a_{k-1}) \simeq a \ .$$

# Definition 4.1

**Example:** Let $\Sigma = \{0, 1, a, b, \sqcup\}$ where $0, 1, a, b, \sqcup$ are different.
- If the tape starting with the head is as follows:
  - $01001\sqcup\sqcup 0101\sqcup\sqcup$
  - or $01001a\sqcup\sqcup$,
  output is $(01001)_2 = 9$.
- If tape starting with the head is as follows:
  - $ab\sqcup\sqcup$
  - or $a$,
  - or $\sqcup\sqcup$,
  the output is $0$.
- **Case 2:** Otherwise.
  Then $\mathrm{T}^{(k)}(a_0, \ldots, a_{k-1})\uparrow$.

# Remark

- If the TM terminates with the head in the middle of a binary number, only the portion of this number starting with the head counts.
- Example: Assume the TM ends as follows
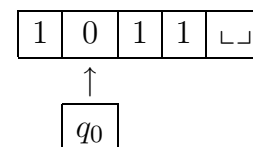


  Then the output is $(011)_2$ which is $3$.

# Definition 4.2

$f : \mathbb{N}^k \xrightarrow{\sim} \mathbb{N}$ *is* **Turing-computable***, in short*
**TM-computable***, if $f = \mathrm{T}^{(k)}$ for some TM $\mathrm{T}$, the alphabet of which contains $\{0, 1\}$.*

**Example:** That $\mathrm{succ} : \mathbb{N} \xrightarrow{\sim} \mathbb{N}$ and $\mathrm{zero} : \mathbb{N} \xrightarrow{\sim} \mathbb{N}$ are Turing-computable was shown above.

# Simpler Solution for zero

- zero can be defined in a simpler way by defining a TM which writes a blank and moves right, then moves back (left) and stops with the head pointing to this blank:
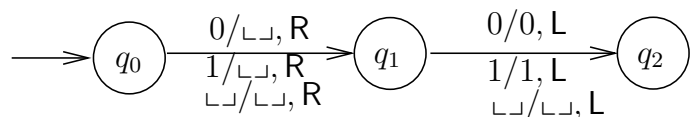


The final state of this TM, run with input some binary number, is as follows ($x$ is $0$, $1$ or $\llcorner\lrcorner$):

# Remark

- If the tape of the Turing machine initially contains only finitely many cells which are not blank, then at any step during the execution of the TM only finitely many cells are non blank.
  - Follows since in each step at most one cell can be modified to become non-blank.
  - So in finitely many steps only finitely many cells can be converted from blank to non-blank.
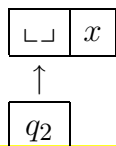
# Simpler Solution for zero

- The output of $\mathrm{T}^{(1)}(x)$ is the value of largest binary string in the final configuration starting with the head position.
- This string is the empty string, which is interpreted as $0$.

# (b) URM Comput. $\Rightarrow$ TM Comput.

**Theorem 4.3** If $f : \mathbb{N}^n \xrightarrow{\sim} \mathbb{N}$ is URM-computable, then it is

as well Turing-computable by a TM with alphabet $\{0, 1, \llcorner\lrcorner\}$.

# Proof of Theorem 4.3

**Notation**
**The tape of a TM contains** $a_0, \ldots, a_l$ means:

- Starting from the head position, the cells of the tape contain $a_0, \ldots, a_l$.
- All other cells contain $\llcorner\lrcorner$.

# Proof of Theorem 4.3

Assume

- $f = \mathrm{U}^{(n)}$,
- $\mathrm{U}$ refers only to $\mathrm{R}_0, \ldots, \mathrm{R}_{l-1}$ and $l > n$,

We define a TM $\mathrm{T}$, which simulates $\mathrm{U}$. Done as follows:

- That the registers $\mathrm{R}_0, \ldots, \mathrm{R}_{l-1}$ contain $a_0, \ldots, a_{l-1}$ is simulated by the tape containing $\widetilde{\mathrm{bin}}(a_0)\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_{l-1})$.
- An instruction $\mathrm{I}_j$ will be simulated by states $s_{j,0}, \ldots, s_{j,i}$ with instructions for those states.

# Conditions on the Simulation

- Assume the URM $\mathrm{U}$ is in a state s.t.
  - $\mathrm{R}_0, \ldots, \mathrm{R}_{l-1}$ contain $a_0, \ldots, a_{l-1}$,
  - the URM is about to execute $\mathrm{I}_j$.
- Assume after executing $\mathrm{I}_j$, the URM is in a state where
  - $\mathrm{R}_0, \ldots, \mathrm{R}_{l-1}$ contain $b_0, \ldots, b_{l-1}$,
  - the PC contains $k$.
- Then we want that, if configuration of the TM $\mathrm{T}$ is, s.t.
  - the tape contains $\widetilde{\mathrm{bin}}(a_0)\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_1)\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_{l-1})$,
  - and the TM is in state $s_{j,0}$,
- then the TM reaches a configuration s.t.
  - the tape contains $\widetilde{\mathrm{bin}}(b_0)\llcorner\lrcorner\widetilde{\mathrm{bin}}(b_1)\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(b_{l-1})$,
  - the TM is in state $s_{k,0}$.

# Example

- Assume the URM is about to execute instruction
  - $\mathrm{I}_4 = \mathtt{pred}(2)$ (i.e. PC = 4),
  - with register contents

    | $\mathrm{R}_0$ | $\mathrm{R}_1$ | $\mathrm{R}_2$ |
    | --- | --- | --- |
    | 2 | 1 | 3 |

- Then the URM will end with
  - PC = 5
  - and register contents

    | $\mathrm{R}_0$ | $\mathrm{R}_1$ | $\mathrm{R}_2$ |
    | --- | --- | --- |
    | 2 | 1 | 2 |

# Example

- Then we want that, if the simulating TM is
  - in state $s_{4,0}$,
  - with tape content $\widetilde{\mathrm{bin}}(2)\llcorner\lrcorner\widetilde{\mathrm{bin}}(1)\llcorner\lrcorner\widetilde{\mathrm{bin}}(3)$
- it should reach
  - state $s_{5,0}$
  - with tape content $\widetilde{\mathrm{bin}}(2)\llcorner\lrcorner\widetilde{\mathrm{bin}}(1)\llcorner\lrcorner\widetilde{\mathrm{bin}}(2)$

# Proof of Theorem 4.3

Assume the run of the URM, starting with $\mathrm{R}_i$ containing $a_{0,i} = a_i$ $i = 0, \ldots, n-1$, and $a_{0,i} = 0$ for $i = n, \ldots, l-1$ is as follows:

| Instruction | $\mathrm{R}_0$ | $\mathrm{R}_1$ | $\cdots$ | $\mathrm{R}_{n-1}$ | $\mathrm{R}_n$ | $\cdots$ | $\mathrm{R}_{l-1}$ |
|---|---|---|---|---|---|---|---|
| $\mathrm{I}_0$ | $a_0$ | $a_1$ | $\cdots$ | $a_{n-1}$ | $0$ | $\cdots$ | $0$ |
| $=$ | $=$ | $=$ | | $=$ | $=$ | | $=$ |
| $\mathrm{I}_{k_0}$ | $a_{0,0}$ | $a_{0,1}$ | $\cdots$ | $a_{0,n-1}$ | $a_{0,n}$ | $\cdots$ | $a_{0,l-1}$ |
| $\mathrm{I}_{k_1}$ | $a_{1,0}$ | $a_{1,1}$ | $\cdots$ | $a_{1,n-1}$ | $a_{1,n}$ | $\cdots$ | $a_{1,l-1}$ |
| $\mathrm{I}_{k_2}$ | $a_{2,0}$ | $a_{2,1}$ | $\cdots$ | $a_{2,n-1}$ | $a_{2,n}$ | $\cdots$ | $a_{2,l-1}$ |

$\cdots$

# Proof of Theorem 4.3

- Furthermore, we need initial states $s_{\mathsf{init},0}, \ldots, s_{\mathsf{init},j}$ and corresponding instructions, s.t.
  - if the TM initially contains

    $$\widetilde{\mathrm{bin}}(b_0)\llcorner\lrcorner\widetilde{\mathrm{bin}}(b_1)\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(b_{n-1})$$

  - it will reach state $s_{0,0}$ with the tape containing

    $$\widetilde{\mathrm{bin}}(b_0)\llcorner\lrcorner\widetilde{\mathrm{bin}}(b_1)\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(b_{n-1})\llcorner\lrcorner\underbrace{0\llcorner\lrcorner0\llcorner\lrcorner\cdots\llcorner\lrcorner0}_{l-n\text{ times}}\llcorner\lrcorner$$

# Proof of Theorem 4.3

Then the corresponding TM will successively reach the following configurations:

| State | Tape contains |
|---|---|
| $s_{\mathsf{init},0}$ | $\widetilde{\mathrm{bin}}(a_0)\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_1)\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_{n-1})\llcorner\lrcorner$ |
| $s_{0,0}$ | $\widetilde{\mathrm{bin}}(a_0)\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_1)\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_{n-1})\llcorner\lrcorner\widetilde{\mathrm{bin}}(0)\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(0$ |
| $=$ | $=$ |
| $s_{k_0,0}$ | $\widetilde{\mathrm{bin}}(a_{0,0})\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_{0,1})\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_{0,l-1})\llcorner\lrcorner$ |
| $s_{k_1,0}$ | $\widetilde{\mathrm{bin}}(a_{1,0})\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_{1,1})\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_{1,l-1})\llcorner\lrcorner$ |
| $s_{k_2,0}$ | $\widetilde{\mathrm{bin}}(a_{2,0})\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_{2,1})\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_{2,l-1})\llcorner\lrcorner$ |

$\cdots$

# Example

Consider the URM program $U$ (which was discussed already in the section on URMs):

$I_0 = \mathtt{ifzero}(0,3)$
$I_1 = \mathtt{pred}(0)$
$I_2 = \mathtt{ifzero}(1,0)$

$U^{(1)}(a) \simeq 0$.

# Corresponding TM Simulation

$I_0 = \mathtt{ifzero}(0,3)$
$I_1 = \mathtt{pred}(0)$
$I_2 = \mathtt{ifzero}(1,0)$

| Instruction | $R_0$ | $R_1$ | State of TM | Content of Tape |
|---|---|---|---|---|
| | | | $s_{\mathsf{init},0}$ | $\widetilde{\mathsf{bin}}(2){\llcorner\lrcorner}$ |
| $I_0$ | 2 | 0 | $s_{0,0}$ | $\widetilde{\mathsf{bin}}(2){\llcorner\lrcorner}\widetilde{\mathsf{bin}}(0){\llcorner\lrcorner}$ |
| $I_1$ | 2 | 0 | $s_{1,0}$ | $\widetilde{\mathsf{bin}}(2){\llcorner\lrcorner}\widetilde{\mathsf{bin}}(0){\llcorner\lrcorner}$ |
| $I_2$ | 1 | 0 | $s_{2,0}$ | $\widetilde{\mathsf{bin}}(1){\llcorner\lrcorner}\widetilde{\mathsf{bin}}(0){\llcorner\lrcorner}$ |
| $I_0$ | 1 | 0 | $s_{0,0}$ | $\widetilde{\mathsf{bin}}(1){\llcorner\lrcorner}\widetilde{\mathsf{bin}}(0){\llcorner\lrcorner}$ |
| $I_1$ | 1 | 0 | $s_{1,0}$ | $\widetilde{\mathsf{bin}}(1){\llcorner\lrcorner}\widetilde{\mathsf{bin}}(0){\llcorner\lrcorner}$ |
| $I_2$ | 0 | 0 | $s_{2,0}$ | $\widetilde{\mathsf{bin}}(0){\llcorner\lrcorner}\widetilde{\mathsf{bin}}(0){\llcorner\lrcorner}$ |
| $I_0$ | 0 | 0 | $s_{0,0}$ | $\widetilde{\mathsf{bin}}(0){\llcorner\lrcorner}\widetilde{\mathsf{bin}}(0){\llcorner\lrcorner}$ |
| $I_3$ | 0 | 0 | $s_{3,0}$ | $\widetilde{\mathsf{bin}}(0){\llcorner\lrcorner}\widetilde{\mathsf{bin}}(0){\llcorner\lrcorner}$ |

URM Stops                                     TM Stops

# Example

$I_0 = \mathtt{ifzero}(0,3)$
$I_1 = \mathtt{pred}(0)$
$I_2 = \mathtt{ifzero}(1,0)$

We saw in the last section that a run of $U^{(1)}(2)$ is as follows:

| Instruction | $R_0$ | $R_1$ |
|---|---|---|
| $I_0$ | 2 | 0 |
| $I_1$ | 2 | 0 |
| $I_2$ | 1 | 0 |
| $I_0$ | 1 | 0 |
| $I_1$ | 1 | 0 |
| $I_2$ | 0 | 0 |
| $I_0$ | 0 | 0 |
| $I_3$ | 0 | 0 |

URM Stops

# Proof of Theorem 4.3

If we have defined this we have

- If
$$U^{(n)}(a_0, \ldots, a_{n-1}) \downarrow \ ,$$
$$U^{(n)}(a_0, \ldots, a_{n-1}) \simeq c \ ,$$

then $U$ eventually stops with $R_i$ containing some values $b_i$, where $b_0 = c$.
Then, the TM $T$ starting with

$$\mathsf{bin}(a_0){\llcorner\lrcorner}\cdots{\llcorner\lrcorner}\mathsf{bin}(a_{n-1})$$

will eventually terminate in a configuration

$$\widetilde{\mathsf{bin}}(b_0){\llcorner\lrcorner}\cdots{\llcorner\lrcorner}\widetilde{\mathsf{bin}}(b_{k-1}) \ .$$

for some $k \geq n$.
Therefore $T^{(n)}(a_0, \ldots, a_{n-1}) \simeq b_0 = c$.

# Proof of Theorem 4.3

- If
$$\mathrm{U}^{(n)}(a_0, \ldots, a_{n-1})\uparrow \ ,$$
  the URM $\mathrm{U}$ will loop and the TM $\mathrm{T}$ will carry out the same steps as the URM and loop as well.
  Therefore
$$\mathrm{T}^{(n)}(a_0, \ldots, a_{n-1})\uparrow \ ,$$
  again
$$\mathrm{U}^{(n)}(a_0, \ldots, a_{n-1}) \simeq \mathrm{T}^{(n)}(a_0, \ldots, a_{n-1}) \ .$$

# Proof of Theorem 4.3

- It follows
$$\mathrm{U}^{(n)} = \mathrm{T}^{(n)} \ ,$$
  and the proof is complete, if the simulation has been introduced.

- The following slides contain a detailed proof, which will not be presented in the lecture this year.
  Jump over remaining proof.

# Proof of Theorem 4.3

Informal description of the simulation of URM instructions.

- **Initialisation.**
  Initially, the tape contains $\mathrm{bin}(a_0)\llcorner\lrcorner\cdots\llcorner\lrcorner\mathrm{bin}(a_{n-1})$.
  We need to obtain configuration:
  $\widetilde{\mathrm{bin}}(a_0)\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(a_{n-1})\llcorner\lrcorner\underbrace{\widetilde{\mathrm{bin}}(0)\llcorner\lrcorner\cdots\llcorner\lrcorner\widetilde{\mathrm{bin}}(0)}_{l-n \text{ times}}$.

  Achieved by
  - moving head to the end of the initial configuration
  - inserting, starting from the next blank, $l - n$-times $0\llcorner\lrcorner$,
  - then moving back to the beginning.

# Proof of Theorem 4.3

- **Simulation of URM instructions.**
  - **Simulation of instruction** $\mathrm{I}_k = \mathrm{succ}(j)$.
    Need to increase $(j+1)$st binary number by $1$  Initial configuration:

  $\widetilde{\mathrm{bin}}(c_0) \quad \llcorner\lrcorner \quad \widetilde{\mathrm{bin}}(c_1) \quad \llcorner\lrcorner \quad \cdots \quad \llcorner\lrcorner \quad \widetilde{\mathrm{bin}}(c_j) \quad \llcorner\lrcorner \quad \cdots \quad \llcorner\lrcorner \quad \widetilde{\mathrm{bin}}(c_i$
  $\uparrow$
  $s_{k,0}$

  - First move to the $(j+1)$st blank to the right. Then we are at the end of the $(j+1)$st binary number.

  $\widetilde{\mathrm{bin}}(c_0) \quad \llcorner\lrcorner \quad \widetilde{\mathrm{bin}}(c_1) \quad \llcorner\lrcorner \quad \cdots \quad \llcorner\lrcorner \quad \widetilde{\mathrm{bin}}(c_j) \quad \llcorner\lrcorner \quad \cdots \quad \llcorner\lrcorner \quad \widetilde{\mathrm{bin}}(c_i$
  $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \uparrow$

# Proof of Theorem 4.3

$$\widetilde{\text{bin}}(c_0) \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_1) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_j) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_l) \quad \sqcup\sqcup$$
$$\uparrow$$

- Now perform the operation for increasing by $1$ as above.
  At the end we obtain:

$$\widetilde{\text{bin}}(c_0) \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_1) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_j + 1) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_l) \quad \sqcup$$
$$\uparrow$$

- It might be that we needed to write over the separating blank a $1$, in which case we have:

$$\widetilde{\text{bin}}(c_0) \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_1) \quad \sqcup\sqcup \quad \cdots \quad \widetilde{\text{bin}}(c_{j-1}) \quad \widetilde{\text{bin}}(c_j + 1) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bir}}$$
$$\uparrow$$

# Proof of Theorem 4.3

- In the latter case, shift all symbols to the left once left, in order to obtain a separating $\sqcup$ between the $l$th and $l - 1$st entry.
  We obtain

$$\widetilde{\text{bin}}(c_0) \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_1) \quad \sqcup\sqcup \quad \cdots \quad \widetilde{\text{bin}}(c_{j-1}) \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_j + 1) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup$$
$$\uparrow$$

- Otherwise, move the head to the left, until we reach the $(j + 1)$st blank to the left, and then move it once to the right.
  We obtain

$$\widetilde{\text{bin}}(c_0) \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_1) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_j + 1) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_l) \quad \sqcup$$
$$\uparrow$$

# Proof of Theorem 4.3

- **Simulation of instruction** $I_k = \text{pred}(j)$.
  - Assume the configuration at the beginning is :

$$\widetilde{\text{bin}}(c_0) \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_1) \quad \sqcup\sqcup \quad \cdots \quad \widetilde{\text{bin}}(c_j) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bi}}$$
$$\uparrow$$

We want to achieve

$$\widetilde{\text{bin}}(c_0) \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_1) \quad \sqcup\sqcup \quad \cdots \quad \widetilde{\text{bin}}(c_j \dot- 1) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup$$
$$\uparrow$$

Done as follows:

# Proof of Theorem 4.3

Initially: $\widetilde{\text{bin}}(c_0) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_j) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bin}}($
$\uparrow$
Finally: $\widetilde{\text{bin}}(c_0) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bin}}(c_j \dot- 1) \quad \sqcup\sqcup \quad \cdots \quad \sqcup\sqcup \quad \widetilde{\text{bin}}($
$\uparrow$

- Move to end of the $(j + 1)$st number.
- Check, if the number consists only of zeros or not.
  - If it consists only of zeros, $\text{pred}(j)$ doesn't change anything.
  - Otherwise, number is of the form $b_0 \cdots b_k 1 \underbrace{00 \cdots 0}_{l' \text{ times}}$.

    Replace it by $b_0 \cdots b_k 0 \underbrace{11 \cdots 1}_{l' \text{ times}}$.

    Done as for $\text{succ}$.

# Proof of Theorem 4.3

Initially: $\widetilde{\mathsf{bin}}(c_0)$ ⊔⌐ $\cdots$ ⊔⌐ $\widetilde{\mathsf{bin}}(c_j)$ ⊔⌐ $\cdots$ ⊔⌐ $\widetilde{\mathsf{bin}}(c_l)$ ⊔⌐
↑

Finally: $\widetilde{\mathsf{bin}}(c_0)$ ⊔⌐ $\cdots$ ⊔⌐ $\widetilde{\mathsf{bin}}(c_j \dotminus 1)$ ⊔⌐ $\cdots$ ⊔⌐ $\widetilde{\mathsf{bin}}(c_l)$ ⊔⌐
↑

We have achieved

$\widetilde{\mathsf{bin}}(c_0)$ ⊔⌐ $\widetilde{\mathsf{bin}}(c_1)$ ⊔⌐ $\cdots$ $\widetilde{\mathsf{bin}}(c_j \dotminus 1)$ ⊔⌐ $\cdots$ ⊔⌐ $\widetilde{\mathsf{bin}}(c_l$
                                        ↑

Move back to the beginning:

$\widetilde{\mathsf{bin}}(c_0)$ ⊔⌐ $\widetilde{\mathsf{bin}}(c_1)$ ⊔⌐ $\cdots$ $\widetilde{\mathsf{bin}}(c_j \dotminus 1)$ ⊔⌐ $\cdots$ ⊔⌐ $\widetilde{\mathsf{bin}}(c_l$
↑

# Remark

- We will later show that all TM-computable functions are URM-computable.
  - This will be done by showing that
    - all TM-computable functions are partial recursive,
    - all partial recursive functions are URM-computable.
  - This will be easier than showing directly that TM-computable functions are URM-computable.
- Therefore the set of TM-computable functions and the set of URM-computable functions coincide.

# Proof of Theorem 4.3

- **Simulation of instruction** $I_k = \mathtt{ifzero}(j, k')$.
  - Move to $j + 1$st binary number on the tape.
  - Check whether it contains only zeros.
    - If yes, switch to state $s_{k',0}$.
    - Otherwise switch to state $s_{k+1,0}$.

This completes the simulation of the URM $U$.

# Extension to Arbitrary Alphabets

- Let $A$ be a finite alphabet s.t. ⊔⌐ $\notin A$, and $B := A^*$.
- To a Turing machine $T = (\Sigma, \mathsf{S}, \mathsf{I}, ⊔⌐, s_0)$ with $A \subseteq \Sigma$ corresponds a partial function $T^{(A,n)} : B^n \xrightarrow{\sim} B$, where $T^{(A,n)}(a_0, \ldots, a_{n-1})$ is computed as follows:
  - Initially write $a_0 ⊔⌐ \cdots ⊔⌐ a_{n-1}$ on the tape, otherwise ⊔⌐.
    Start in state $s_0$ on the left most position of $a_0$.
  - Iterate TM as before.
  - In case of termination, the output of the function is $c_0 \cdots c_{l-1}$, if the tape contains, starting with the head position $c_0 \cdots c_{l-1}d$ with $c_i \in A$, $d \notin A$.
  - Otherwise, the function value is undefined.

# Characteristic function

- In order to introduce the notion of Turing-decidable we need to remind us of the following definition:

- Let $M \subseteq \mathbb{N}^n$ be a predicate. The characteristic function $\chi_M : \mathbb{N}^n \to \mathbb{N}$ for $M$ is defined as follows:

$$\chi_M(\vec{x}) := \begin{cases} 1 & \text{if } M(\vec{x}) \text{ holds,} \\ 0 & \text{otherwise} \end{cases}$$

- If we treat true as $1$ and false as $0$, then the characteristic function is nothing but the Boolean valued function which decides whether $M(\vec{x})$ holds or not:

$$\chi_M(\vec{x}) = \begin{cases} \text{true} & \text{if } M(\vec{x}) \text{ holds,} \\ \text{false} & \text{otherwise} \end{cases}$$

# Extension to Arbitrary Alphabets

- Notion is modulo encoding of $A^*$ into $\mathbb{N}$ equivalent to the notion of Turing-computability on $\mathbb{N}$.

- However, when considering complexity bounds, this notation might be more appropriate.
  - Avoids encoding/decoding into $\mathbb{N}$.

# Turing-Computable Predicates

- A predicate $A$ is Turing-decidable, iff $\chi_A$ is Turing-computable.

- Instead of simulating $\chi_A$
  - means to write the output of $\chi_A$ (a binary number $0$ or $1$) on the tape

  it is more convenient, to take TM with two additional special states $s_{\text{true}}$ and $s_{\text{false}}$ corresponding to truth and falsity of the predicate.

# Turing-Computable Predicates

- Then a predicate is Turing decidable, if, when we write initially the inputs as before on the tape and start executing the TM,
  - it always terminates in $s_{\text{true}}$ or $s_{\text{false}}$,
  - and it terminates in $s_{\text{true}}$, iff the predicate holds for the inputs,
  - and in $s_{\text{false}}$, otherwise.

- The latter notion is equivalent to the first notion.

- Usually the latter one is taken as basis for complexity considerations.

# (c) Undecid. of the Halting Problem

- Undecidability of the Halting Problem first proved 1936 by Alan Turing.
- In this Section, we will identify computable with Turing-computable.
  - This will later be justified by the Church-Turing thesis.

# Definition 4.4

(a) A **problem** is an $n$-ary predicate $M(\vec{x})$ of natural numbers, i.e. a property of n-tuples of natural numbers.

(b) A problem (or predicate) $M$ is (Turing-)**decidable**, if **the characteristic function $\chi_M$ of $M$** is (Turing-)computable.

# History of Computability Theory



**Alan Mathison Turing (1912 – 1954)**
Introduced the Turing machine.
Proved the undecidability
of the Turing-Halting problem.

# Example of Decidable Problems

- The binary predicate

$$\mathrm{Multiple}(x, y) :\Leftrightarrow x \text{ is a multiple of } y$$

  is a predicate and therefore a problem.

- $\chi_{\mathrm{Multiple}}(x, y)$ decides, whether $\mathrm{Multiple}(x, y)$ holds (then it returns $1$ for yes), or not:

$$\chi_{\mathrm{Multiple}}(x, y) = \begin{cases} 1 & \text{if } x \text{ is a multiple of } y, \\ 0 & \text{if } x \text{ is not a multiple of } y. \end{cases}$$

- $\chi_{\mathrm{Multiple}}$ is intuitively computable, therefore $\mathrm{Multiple}$ is decidable.

# Need of Encoding of TMs

- We want to show that it is not decidable whether a Turing Machine terminates or not.
- For this we need to be able to talk about programs which have as input a Turing Machine.
- For this we need to give a formalisation of what a Turing Machine is.
- Since we are restricting ourselves to functions having as arguments elements of $\mathbb{N}^k$, we need to encode a TM as an element of $\mathbb{N}^k$ for some $k$.
- We will actually encode TMs as elements of $\mathbb{N}$.

# Encoding of Turing Machines

- Let for a Turing machine $\mathrm{T}$, $\mathrm{encode}(\mathrm{T}) \in \mathbb{N}$ be its code.
- It is intuitively decidable, whether a string of ASCII symbols is a Turing machine.
  - One can show that this can be decided by a Turing machine.
- $\Rightarrow$ It is intuitively decidable, whether $n = \mathrm{encode}(\mathrm{T})$ for a Turing machine $\mathrm{T}$.

# Encoding of Turing Machines

- A Turing Machine is a quintuple $(\Sigma, \mathsf{S}, \mathsf{I}, \llcorner\lrcorner, s_0)$.
- We can assume that $\llcorner\lrcorner$, each symbol of the alphabet, and each state can be represented by a string of letters and numbers.
- Then this quintuple can be written as a string of ASCII-symbols.
- $\Rightarrow$ Turing machines can be represented as elements of $A^*$, where
  $A =$ set of ASCII-symbols.
- $\Rightarrow$ Turing machines can be encoded as natural numbers.
  - Of course more efficient encoding exist.

# $\{e\}^k(n)$

- Assume $e \in \mathbb{N}$. We define a partial function $\{e\}^k : \mathbb{N}^k \xrightarrow{\sim} \mathbb{N}$, by

$$\{e\}^k(\vec{x}) \simeq \begin{cases} m & \text{if } e = \mathrm{encode}(\mathrm{T}) \text{ for some Turing machir} \\ & \quad \text{and } \mathrm{T}^{(k)}(\vec{x}) \simeq m, \\ \bot & \text{otherwise.} \end{cases}$$

- So if $e = \mathrm{encode}(\mathrm{T})$, $\{e\}^k = \mathrm{T}^{(k)}$.
  - Roughly speaking, $\{e\}^k$ is the function computed by the $e$th Turing machine.
  - So for every computable (more precisely Turing-computable) function $f : \mathbb{N}^k \xrightarrow{\sim} \mathbb{N}$ there exists an $e$ s.t. $f = \{e\}^k$.

# $\{e\}^k$

- The notation $\{e\}^k$ is due to Stephen Kleene.
- $\{\}$ are called **Kleene-Brackets**.
- We write $\{e\}$ for $\{e\}^1$.

# Stephen Cole Kleene



**Stephen Cole Kleene (1909 – 1994)**
Probably the most influential computability theorist up to now. Introduced the partial recursive functions.

# The Halting Problem

- **Definition 4.5**
  The **Halting Problem** is the following binary predicate:

$$\mathrm{Halt}(e, n) :\Leftrightarrow \{e\}(n)\downarrow$$

- We will show that $\mathrm{Halt}$ is undecdiable.

# Example

- Let $e = \mathrm{encode}(\mathrm{T})$, where $\mathrm{T}$ is the Turing machine $\mathrm{T}$ which translates the URM program consisting of only one instruction

$$\mathrm{I}_0 = \mathtt{ifzero}(0,0)$$

- If this TM is run with arguments written on the tape, it loops if the first argument is $0$, and terminates otherwise with its first argument unchanged.
- So we have

$$\{e\}(k) \simeq \mathrm{T}^{(1)}(k) \simeq \left\{ \begin{array}{ll} k & \text{if } k > 0 \\ \bot & \text{otherwise.} \end{array} \right.$$

- Therefore $\mathrm{Halt}(e, k)$ holds for $k > 0$ and does not hold for $k = 0$.

# Remark

- Below we will see: Halt is undecideable.

- However, the following function WeakHalt is computable:

$$\text{WeakHalt}(e, n) :\simeq \begin{cases} 1 & \text{if } \{e\}(n)\downarrow \\ \bot & \text{otherwise} \end{cases}$$

- Computed as follows:
  First check whether $e = \text{encode}(\text{T})$ for some Turing machine T.
  If not, enter an infinite loop.
  Otherwise, simulate T with input $n$.
  If simulation stops, output $1$, otherwise the program loops for ever.

# Question

- What is WeakHalt$(e, n)$, where $e$ is a code for the Turing machine corresponding to the URM program

$$\text{I}_0 = \texttt{ifzero}(0,0) \ ?$$

# Theorem 4.6

**Theorem 4.6** *The halting problem is undecidable.*

**Proof:**

- **Assume** the Halting problem is decidable
  i.e. assume that we can decide using a Turing machine whether $\{e\}(n)\downarrow$ holds.

- We will define below a computable function $f : \mathbb{N} \xrightarrow{\sim} \mathbb{N}$, s.t. $f \neq \{e\}$.

- Therefore $f$ cannot be computed by the Turing machine with code $e$ for any $e$, i.e. $f$ is noncomputable.

- Therefore we obtain a **contradiction.**

# Proof of Theorem 4.6

- We argue similarly as in the proof of $\mathbb{N} \not\approx \mathcal{P}(\mathbb{N})$
  - We define $f(e)$ in such a way that $f = \{e\}$ is violated by having $f(e) \not\simeq \{e\}(e)$.
- If $\{e\}(e)\downarrow$, then we let $f(e)\uparrow$.
- If $\{e\}(e)\uparrow$, we let $f(e)\downarrow$, e.g. by defining $f(e) \simeq 0$ (any other defined result would be appropriate as well).
- So we define

$$f(e) \simeq \begin{cases} \bot, & \text{if } \{e\}(e)\downarrow \\ 0, & \text{if } \{e\}(e)\uparrow \end{cases}$$

# Proof of Theorem 4.6

$$f(e) \simeq \begin{cases} \bot, & \text{if } \{e\}(e)\downarrow \\ 0, & \text{if } \{e\}(e)\uparrow \end{cases}$$

- We obtain $f(e)\downarrow \Leftrightarrow \{e\}(e)\uparrow$. $\hspace{1em}$ ($*$)

- Since Halt is decidable, $f$ is computable (Exercise: show that $f$ is computable by a Turing machine, assuming a Turing machine for Halt).

- Therefore $f = \{e\}$ for some $e$.

- But then by ($*$)

$$f(e)\downarrow \overset{(*)}{\Leftrightarrow} \{e\}(e)\uparrow \overset{f=\{e\}}{\Leftrightarrow} f(e)\uparrow$$

a contradiction.

# Proof of Theorem 4.6

The complete proof on one slide is as follows:

- Assume Halt were decidable.

- Define

$$f(e) \simeq \begin{cases} \bot, & \text{if } \{e\}(e)\downarrow \\ 0, & \text{if } \{e\}(e)\uparrow \end{cases}$$

- By Halt decidable, we obtain $f$ is computable, so $f = \{e\}$ for some $e$.

- But then

$$f(e)\downarrow \overset{\text{Def of } f}{\Leftrightarrow} \{e\}(e)\uparrow \overset{f=\{e\}}{\Leftrightarrow} f(e)\uparrow$$

# Remark

- The above proof can easily be adapted to any reasonable programming language, in which one can define all intuitively computable functions.

- Such programming languages are called **Turing-complete** languages.
  - Babbage's machine was, if one removes the restriction to finite memory, Turing-complete, since it had a conditional jump.

- For standard Turing complete languages, the unsolvability of the Turing-halting problem means: it is not possible to write a program, which checks, whether a program on given input terminates.

# Halting Problem with no Inputs

- **Theorem 4.7:** It is undecidable, whether a Turing machine started with a blank tape terminates.

- **Proof:**
  - Let

    $\text{Halt}'(e) :\Leftrightarrow e$ is a code for a Turing machine $T$
    $\hspace{3em}$ and $T$ started with a blank tape
    $\hspace{3em}$ terminates

  - Assume $\text{Halt}'$ were decidable.

# Halting Problem with no Inputs

- Then we can decide $\text{Halt}(e, n)$ as follows:
    - Assume inputs $e$, $n$.
    - If $e$ is not a code for a Turing machine, we return $0$.
    - Otherwise, let $\text{encode}(\text{T}) = e$.
    - Define a Turing machine $\text{V}$ as follows:
        - $\text{V}$ first writes $\text{bin}(n)$ on the tape and moves head to the left most bit of $\text{bin}(n)$.
        - Then it executes the Turing machine $\text{T}$.
    - We have
        - $\text{V}$, run with blank tape, terminates
        
        iff $\text{T}$ run with tape containing $\text{bin}(n)$ terminates
        
        iff $\text{T}^{(1)}(n)\downarrow$
        
        iff $\{e\}(n)\downarrow$.

# Halting Problem with no Inputs

$\text{V}$, run with blank tape, terminates iff $\{e\}(n)\downarrow$.

- Let $\text{encode}(\text{V}) = e'$. Then

$$\text{Halt}'(e') \Leftrightarrow \text{Halt}(e, n)$$

- Therefore using the decidability of $\text{Halt}'$ we can decide $\text{Halt}(e, n)$.
- So we have decided $\text{Halt}$, a contradiction.