CS_275 Automata and Formal Language Theory
Course Notes
Additional Material
Part IV: Limits of Computation
Chapt. IV.3: Turing Machines

Anton Setzer

http://www.cs.swan.ac.uk/~csetzer/lectures/automataFormalLanguage/12/index.html

April 14, 2013
IV.3 (a) Definition of the Turing Machine

IV.3 (b) Equivalence of URM computable and Turing computable functions

IV.3 (c) Undecidability of the Turing Halting Problem

IV.4 The Church-Turing Thesis
No Additional Material

For this subsection no additional material has been added yet.
IV.3 (a) Definition of the Turing Machine

IV.3 (b) Equivalence of URM computable and Turing computable functions

IV.3 (c) Undecidability of the Turing Halting Problem

IV.4 The Church-Turing Thesis
Formal Lemma URM-computable $\Rightarrow$ TM-computable

Lemma (3.4)

If $f : \mathbb{N}^n \sim \rightarrow \mathbb{N}$ is URM-computable then it is Turing-computable by a TM with alphabet $\{0, 1, \perp\}$.

Remark

The proof that every Turing computable function is URM computable will not be given in this Section. (It could be done directly. A much nicer argument which makes use of the notion of partial recursive functions can be found in the notes of “Computability Theory”).
Notation: $\tilde{\text{bin}}$

- In this proof we will represent a configuration of a URM by a sequence of possibly non-normalised strings on the tape representing the registers.
- So we want to get a short notation for “The tape contains $s_0 \| \ s_1 \| \ s_2 \| \cdot \cdot \cdot \| \ s_k$ where $s_i$ is a binary representation of $n_i$” (where $n_i$ is the simulated content of register $R_i$).
- We define $\tilde{\text{bin}}(n)$ as one of the binary representations of $s$.
- Then we can write for the above: “The tape contains $\tilde{\text{bin}}(n_0) \| \tilde{\text{bin}}(n_1) \| \cdot \cdot \cdot \| \tilde{\text{bin}}(n_k)$”.
- So $\tilde{\text{bin}}(n)$ denotes one of the possible choices for strings $s$ s.t. $(s)_2 = n$.
  - So $\tilde{\text{bin}}(1)$ can be ”1”, ”01”, ”001”, etc.
  - In the special case 0 we treat the empty string as one of the possible representations, so $\tilde{\text{bin}}(0)$ can be ”” ”0”, ”00”, ”000”, etc.
Notation: $\tilde{\text{bin}}$

- When carrying out intermediate calculations, it is easier to refer to $\tilde{\text{bin}}(n)$ rather than $\text{bin}(n)$
  - E.g. we can set a number on the tape easily to an element of $\tilde{\text{bin}}(0)$ by overwriting it with 0s.
  - 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.
Proof of Lemma 3.4

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 \( \sqcup \).
Proof of Lemma 3.4

Assume

- \( f = U^{(n)} \),
- \( U \) refers only to \( R_0, \ldots, R_{l-1} \) and \( l > n \),

We define a TM \( T \), which simulates \( U \). Done as follows:

- That the registers \( R_0, \ldots, R_{l-1} \) contain \( a_0, \ldots, a_{l-1} \) is simulated by the tape containing \( \text{bin}(a_0) \downarrow \cdots \downarrow \text{bin}(a_{l-1}) \).
- An instruction \( I_j \) will be simulated by states \( s_{j,0}, \ldots, s_{j,i} \) with instructions for those states.
Conditions on the Simulation

- Assume the URM $U$ is in a state s.t.
  - $R_0, \ldots, R_{l-1}$ contain $a_0, \ldots, a_{l-1}$,
  - the URM is about to execute $I_j$.
- Assume after executing $I_j$, the URM is in a state where
  - $R_0, \ldots, R_{l-1}$ contain $b_0, \ldots, b_{l-1}$,
  - the PC contains $k$.
- Then we want that, if configuration of the TM $T$ is, s.t.
  - the tape contains $\text{bin}(a_0)\;\text{bin}(a_1)\;\cdots\;\text{bin}(a_{l-1})$,
  - and the TM is in state $s_{j,0}$,
- then the TM reaches a configuration s.t.
  - the tape contains $\text{bin}(b_0)\;\text{bin}(b_1)\;\cdots\;\text{bin}(b_{l-1})$,
  - the TM is in state $s_{k,0}$. 
Example

- Assume the URM is about to execute instruction
  - 4 : \( R_2 := R_2 \div 1 \) (i.e. \( PC = 4 \)),
  - with register contents
    \[
    \begin{array}{c|c|c}
    R_0 & R_1 & R_2 \\
    \hline
    2 & 1 & 3 \\
    \end{array}
    \]
- Then the URM will end with
  - \( PC = 5 \)
  - and register contents
    \[
    \begin{array}{c|c|c}
    R_0 & R_1 & R_2 \\
    \hline
    2 & 1 & 2 \\
    \end{array}
    \]
Example

Then we want that, if the simulating TM is in state $s_{4,0}$,

- with tape content $\text{bin}(2)\underline{\text{bin}}(1)\underline{\text{bin}}(3)$

it should reach

- state $s_{5,0}$
- with tape content $\text{bin}(2)\underline{\text{bin}}(1)\underline{\text{bin}}(2)$
Proof of Lemma 3.4

- Furthermore, we need initial states $s_{\text{init},0}, \ldots, s_{\text{init},j}$ and corresponding instructions, s.t.
  - if the TM initially contains
    $$\overline{\text{bin}}(b_0) \overline{\text{bin}}(b_1) \cdots \overline{\text{bin}}(b_{n-1})$$
  - it will reach state $s_{0,0}$ with the tape containing
    $$\overline{\text{bin}}(b_0) \overline{\text{bin}}(b_1) \cdots \overline{\text{bin}}(b_{n-1}) \underbrace{000\cdots0}_{l-n \text{ times}}$$
Proof of Lemma 3.4

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

\[
\begin{array}{cccccccc}
\text{Instruction} & R_0 & R_1 & \cdots & R_{n-1} & R_n & \cdots & R_{l-1} \\
0 & a_0 & a_1 & \cdots & a_{n-1} & 0 & \cdots & 0 \\
= & = & = & = & = & = & = & = \\
k_0 & a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} & a_{0,n} & \cdots & a_{0,l-1} \\
k_1 & a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} & a_{1,n} & \cdots & a_{1,l-1} \\
k_2 & a_{2,0} & a_{2,1} & \cdots & a_{2,n-1} & a_{2,n} & \cdots & a_{2,l-1} \\
\cdots & & & & & & &
\end{array}
\]
Proof of Lemma 3.4

Then the corresponding TM will successively reach the following configurations:

<table>
<thead>
<tr>
<th>State</th>
<th>Tape contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_{init,0}$</td>
<td>$\text{bin}(a_0) \overline{\text{bin}}(a_1) \overline{\text{bin}}(a_2) \cdots \overline{\text{bin}}(a_{n-1}) \overline{\text{bin}}(0) \overline{\text{bin}}(1) \cdots \overline{\text{bin}}(1)$</td>
</tr>
<tr>
<td>$s_{0,0}$</td>
<td>$\text{bin}(a_0) \overline{\text{bin}}(a_1) \overline{\text{bin}}(a_2) \cdots \overline{\text{bin}}(a_{n-1}) \overline{\text{bin}}(0) \overline{\text{bin}}(0) \overline{\text{bin}}(0) \overline{\text{bin}}(1) \cdots \overline{\text{bin}}(1)$</td>
</tr>
<tr>
<td>$s_{k_0,0}$</td>
<td>$\text{bin}(a_0,0) \overline{\text{bin}}(a_0,1) \overline{\text{bin}}(a_1,1) \overline{\text{bin}}(a_2,1) \cdots \overline{\text{bin}}(a_0,1-l) \overline{\text{bin}}(a_0,1-l) \cdots \overline{\text{bin}}(a_0,1-l) \overline{\text{bin}}(0) \overline{\text{bin}}(1) \cdots \overline{\text{bin}}(1)$</td>
</tr>
<tr>
<td>$s_{k_1,0}$</td>
<td>$\text{bin}(a_1,0) \overline{\text{bin}}(a_1,1) \overline{\text{bin}}(a_2,1) \cdots \overline{\text{bin}}(a_1,1-l) \overline{\text{bin}}(a_1,1-l) \cdots \overline{\text{bin}}(a_1,1-l) \overline{\text{bin}}(0) \overline{\text{bin}}(1) \cdots \overline{\text{bin}}(1)$</td>
</tr>
<tr>
<td>$s_{k_2,0}$</td>
<td>$\text{bin}(a_2,0) \overline{\text{bin}}(a_2,1) \overline{\text{bin}}(a_3,1) \cdots \overline{\text{bin}}(a_2,1-l) \overline{\text{bin}}(a_2,1-l) \cdots \overline{\text{bin}}(a_2,1-l) \overline{\text{bin}}(0) \overline{\text{bin}}(1) \cdots \overline{\text{bin}}(1)$</td>
</tr>
<tr>
<td></td>
<td>$\cdots$</td>
</tr>
</tbody>
</table>
Example

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

0: if $R_0 = 0$ then goto 3
1: $R_0 := R_0 - 1$
2: if $R_1 = 0$ then goto 0
$U^{(1)}(a) \simeq 0$. 
Example

0: if $R_0 = 0$ then goto 3
1: $R_0 := R_0 \div 1$
2: if $R_1 = 0$ then goto 0

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>$R_0$</th>
<th>$R_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

URM Stops
IV.3 (b) Equivalence of URM computable and Turing computable functions

Corresponding TM Simulation

\[ \text{0: if } R_0 = 0 \text{ then goto 3} \]
\[ \text{1: } R_0 := R_0 \div 1 \]
\[ \text{2: if } R_1 = 0 \text{ then goto 0} \]

<table>
<thead>
<tr>
<th>Instruction</th>
<th>( R_0 )</th>
<th>( R_1 )</th>
<th>State of TM</th>
<th>Content of Tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>( s_{\text{init},0} )</td>
<td>( \text{bin}(2) )</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>( s_{0,0} )</td>
<td>( \text{bin}(2) \text{ bin}(0) )</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>( s_{1,0} )</td>
<td>( \text{bin}(2) \text{ bin}(0) )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( s_{2,0} )</td>
<td>( \text{bin}(1) \text{ bin}(0) )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( s_{0,0} )</td>
<td>( \text{bin}(1) \text{ bin}(0) )</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>( s_{1,0} )</td>
<td>( \text{bin}(1) \text{ bin}(0) )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( s_{2,0} )</td>
<td>( \text{bin}(0) \text{ bin}(0) )</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>( s_{0,0} )</td>
<td>( \text{bin}(0) \text{ bin}(0) )</td>
</tr>
</tbody>
</table>

URM Stops

TM Stops
Proof of Lemma 3.4

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

\[ \text{bin}(a_0) \downarrow \cdots \downarrow \text{bin}(a_{n-1}) \]

will eventually terminate in a configuration

\[ \tilde{\text{bin}}(b_0) \downarrow \cdots \downarrow \tilde{\text{bin}}(b_{k-1}) \]

for some \( k \geq n \).

Therefore \( T^{(n)}(a_0, \ldots, a_{n-1}) \simeq b_0 = c \).
Proof of Lemma 3.4

If

\[ U^{(n)}(a_0, \ldots, a_{n-1}) \uparrow, \]

the URM \( U \) will loop and the TM \( T \) will carry out the same steps as the URM and loop as well.

Therefore

\[ T^{(n)}(a_0, \ldots, a_{n-1}) \uparrow, \]

again

\[ U^{(n)}(a_0, \ldots, a_{n-1}) \simeq T^{(n)}(a_0, \ldots, a_{n-1}). \]
Proof of Lemma 3.4

- It follows

\[ U^{(n)} = T^{(n)} , \]

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 Lemma 3.4

Informal description of the simulation of URM instructions.

▷ **Initialisation.**

Initially, the tape contains bin$(a_0)\underline{\ldots}\underline{\ldots}\underline{\ldots}$bin$(a_{n-1})$. We need to obtain configuration:
\[
\overline{\text{bin}(a_0)\underline{\ldots}\underline{\ldots}\underline{\ldots}\text{bin}(a_{n-1})\underline{\ldots}\underline{\ldots}\underline{\ldots}\text{bin}(0)\underline{\ldots}\underline{\ldots}\underline{\ldots}\text{bin}(0)}.
\]

Achieved by

▷ moving head to the end of the initial configuration
▷ inserting, starting from the next blank, $l - n$-times 0\underline{\ldots},
▷ then moving back to the beginning.
Proof of Lemma 3.4

- **Simulation of URM instructions.**

- **Simulation of instruction** $k : R_j := R_j + 1$.

  Need to increase $(j + 1)$st binary number by 1  
  Initial configuration:

  $\begin{aligned}
  \widetilde{\text{bin}}(c_0) & \quad \widetilde{\text{bin}}(c_1) & \quad \cdots & \quad \widetilde{\text{bin}}(c_j) & \quad \cdots & \quad \widetilde{\text{bin}}(c_l) \\
  \uparrow \quad s_{k,0}
  \end{aligned}$

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

  $\begin{aligned}
  \widetilde{\text{bin}}(c_0) & \quad \widetilde{\text{bin}}(c_1) & \quad \cdots & \quad \widetilde{\text{bin}}(c_j) & \quad \cdots & \quad \widetilde{\text{bin}}(c_l) \\
  \uparrow
  \end{aligned}$
Proof of Lemma 3.4

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

\[ \tilde{\text{bin}}(c_0) \quad \tilde{\text{bin}}(c_1) \quad \cdots \quad \tilde{\text{bin}}(c_j) \quad \cdots \quad \tilde{\text{bin}}(c_l) \]

\[ \uparrow \]

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

\[ \tilde{\text{bin}}(c_0) \quad \tilde{\text{bin}}(c_1) \quad \cdots \quad \tilde{\text{bin}}(c_j - 1) \quad \tilde{\text{bin}}(c_j + 1) \quad \cdots \quad \tilde{\text{bin}}(c_l) \]

\[ \uparrow \]
Proof of Lemma 3.4

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

  We obtain
  \[
  \tilde{\text{bin}}(c_0) \Box \tilde{\text{bin}}(c_1) \Box \cdots \tilde{\text{bin}}(c_{j-1}) \Box \tilde{\text{bin}}(c_j + 1) \Box \cdots \Box
  \]

- 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
  \[
  \tilde{\text{bin}}(c_0) \Box \tilde{\text{bin}}(c_1) \Box \cdots \Box \tilde{\text{bin}}(c_j + 1) \Box \cdots \Box \tilde{\text{bin}}(c_l) \Box
  \]
Proof of Lemma 3.4

**Simulation of instruction** $k : R_j := R_j - 1$.

- Assume the configuration at the beginning is:

  $$\begin{array}{c}
  \tilde{\text{bin}}(c_0) \quad \tilde{\text{bin}}(c_1) \quad \cdots \quad \tilde{\text{bin}}(c_j) \quad \cdots \quad \tilde{\text{bin}}(c_l)
  \end{array}$$

  \[\uparrow\]

  We want to achieve

  $$\begin{array}{c}
  \tilde{\text{bin}}(c_0) \quad \tilde{\text{bin}}(c_1) \quad \cdots \quad \tilde{\text{bin}}(c_j - 1) \quad \cdots \quad \tilde{\text{bin}}(c_l)
  \end{array}$$

  \[\uparrow\]

  Done as follows:
Proof of Lemma 3.4

Initially: \( \overline{\text{bin}}(c_0) \quad \cdots \quad \overline{\text{bin}}(c_j) \quad \cdots \quad \overline{\text{bin}}(c_l) \)

\[ \uparrow \]

Finally: \( \overline{\text{bin}}(c_0) \quad \cdots \quad \overline{\text{bin}}(c_j - 1) \quad \cdots \quad \overline{\text{bin}}(c_l) \)

\[ \uparrow \]

- 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, \( R_j := R_j - 1 \) 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 \( R_j := R_j + 1 \).
Proof of Lemma 3.4

Proof of Lemma 3.4

- **Simulation of instruction** $k : \text{if } R_j = 0 \text{ then goto } k'$.
  - 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 $U$. 
IV.3 (a) Definition of the Turing Machine

IV.3 (b) Equivalence of URM computable and Turing computable functions

IV.3 (c) Undecidability of the Turing Halting Problem

IV.4 The Church-Turing Thesis
Theorem (3.8)

It is undecidable, whether a Turing machine started with a blank tape terminates.

Proof:

Let

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

Assume Halt' were decidable.
Halting Problem with no Inputs

Then we can decide Halt(e, n) as follows:

- Assume inputs e, n.
- If e is not a code for a Turing machine, we return 0.
- Otherwise, let encode(T) = e.
- Define a Turing machine V as follows:
  - V first writes bin(n) on the tape and moves head to the left most bit of bin(n).
  - Then it executes the Turing machine T.

We have

- V, run with blank tape, terminates
  - iff T run with tape containing bin(n) terminates
  - iff T^{(1)}(n) ↓
  - iff \{e\}(n) ↓.
Halting Problem with no Inputs

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

- Let encode(V) = \( e' \). Then
  
  \[
  \text{Halt}'(e') \iff \text{Halt}(e, n)
  \]

- Therefore using the decidability of Halt' we can decide Halt(\( e, n \)).
- So we have decided Halt, a contradiction.
IV.3 (a) Definition of the Turing Machine

IV.3 (b) Equivalence of URM computable and Turing computable functions

IV.3 (c) Undecidability of the Turing Halting Problem

IV.4 The Church-Turing Thesis
For this subsection no additional material has been added yet.