Fundamentele Informatica 3

voorjaar 2015

`http://www.liacs.leidenuniv.nl/~vlietrvan1/fi3/`

Rudy van Vliet
kamer 124 Snellius, tel. 071-527 5777
rvvliet(at)liacs(dot)nl

college 4, 23 februari 2015

7. Turing Machines

7.6. The Church-Turing Thesis
7.7. Nondeterministic Turing Machines
7.8. Universal Turing Machines

7.6. The Church-Turing Thesis

Turing machine is general model of computation.

Any algorithmic procedure that can be carried out at all
(by human computer, team of humans, electronic computer)
can be carried out by a TM.
(Alonzo Church, 1930s)

Evidence for Church-Turing thesis:

1. Nature of the model.

2. Various enhancements of TM do not change computing power.

3. Other theoretical models of computation have been proposed. Various notational systems have been suggested as ways of describing computations. All of them equivalent to TM.

4. No one has suggested any type of computation that ought to be considered ‘algorithmic procedure’ and cannot be implemented on TM.

Once we adopt Church-Turing thesis,

- we have definition of algorithmic procedure
- we may omit details of TMs

7.7. Nondeterministic Turing Machines

A slide from lecture 2

Definition 7.1. Turing machines

A Turing machine (TM) is a 5-tuple $T = (Q, \Sigma, \Gamma, q_0, \delta)$, where

Q is a finite set of states. The two *halt* states h_a and h_r are not elements of Q.

Σ, the input alphabet, and Γ, the tape alphabet, are both finite sets, with $\Sigma \subseteq \Gamma$. The *blank* symbol Δ is not an element of Γ.

q_0, the initial state, is an element of Q.

δ is the transition function:

$$\delta : Q \times (\Gamma \cup \{\Delta\}) \to (Q \cup \{h_a, h_r\}) \times (\Gamma \cup \{\Delta\}) \times \{R, L, S\}$$

Nondeterministic Turing machine.

There may be more than one move for a state-symbol pair.

Same notation:

$$wpax \vdash_T yqbz \qquad wpax \vdash_T^* yqbz$$

A string x is accepted by T if

$$q_0 \Delta x \vdash_T^* wh_a y$$

for some strings $w, y \in (\Gamma \cup \{\Delta\})^*$.

NTM useful for accepting languages, for producing output, but not for computing function.

Example 7.28. The Set of Composite Natural Numbers.

Use *G2*

Example 7.28. The Set of Composite Natural Numbers.

$$\mathit{NB} \to \mathit{G2} \to \mathit{NB} \to \mathit{G2} \to \mathit{PB} \to \mathit{M} \to \mathit{PB} \to \mathit{Equal}$$

Take $x = 1^{15}$

Example 7.30. The Language of Prefixes of Elements of L.

Let $L = L(T)$. Then

$$P(L) = \{x \in \Sigma^* \mid \ xy \in L \text{ for some } y \in \Sigma^*\}$$

Example 7.30. The Language of Prefixes of Elements of L.

Let $L = L(T)$. Then

$$P(L) = \{x \in \Sigma^* \mid \ xy \in L \text{ for some } y \in \Sigma^*\}$$

Deterministic TM accepting $P(L)$ may execute following algorithm for input x:

```
$y = \Lambda$;
while ($T$ does not accept $xy$)
    $y$ is next string in $\Sigma^*$ (in canonical order);
accept;
```

but. . .

Example 7.30. The Language of Prefixes of Elements of L.

Let $L = L(T)$. Then

$$P(L) = \{x \in \Sigma^* \mid \; xy \in L \text{ for some } y \in \Sigma^*\}$$

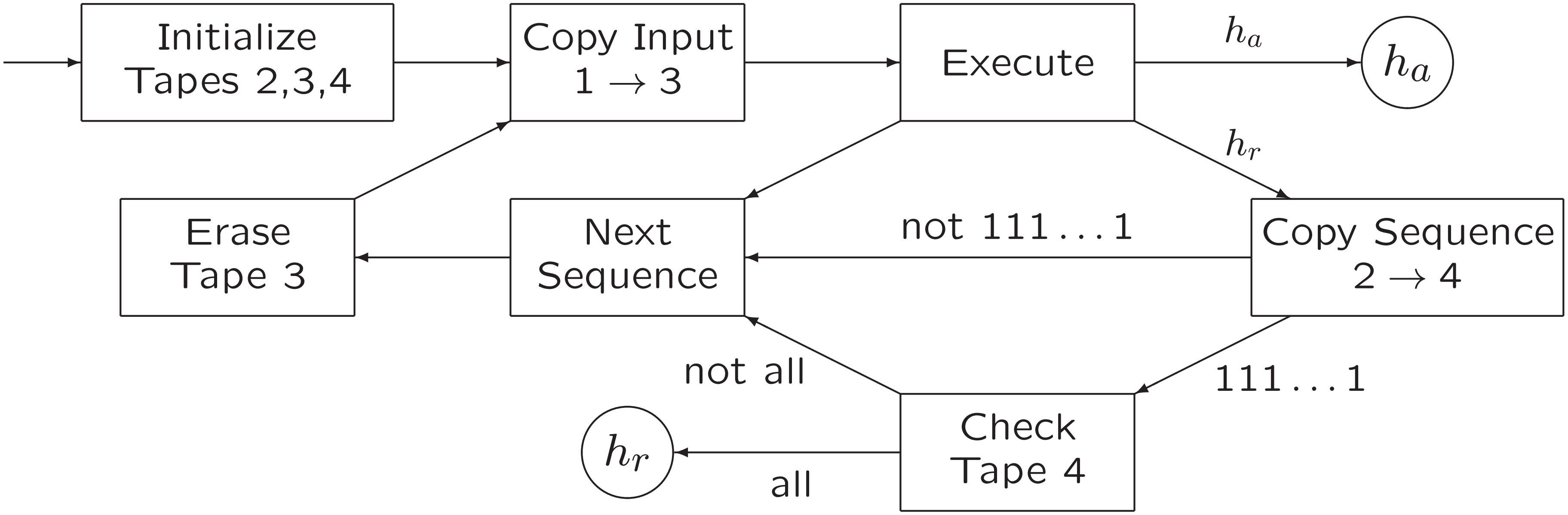
$$\textit{NB} \to \textit{G} \to \textit{Delete} \to \textit{PB} \to T$$

Theorem 7.31.
For every nondeterministic TM $T = (Q, \Sigma, \Gamma, q_0, \delta)$,
there is an ordinary (deterministic) TM $T_1 = (Q_1, \Sigma, \Gamma_1, q_1, \delta_1)$
with $L(T_1) = L(T)$.

Proof. . .

Theorem 7.31.
For every nondeterministic TM $T = (Q, \Sigma, \Gamma, q_0, \delta)$, there is an ordinary (deterministic) TM $T_1 = (Q_1, \Sigma, \Gamma_1, q_1, \delta_1)$ with $L(T_1) = L(T)$.

Proof. . .


Initialize Tapes 2,3,4

Copy Input $1 \to 3$

Execute

h_a

h_r

Erase Tape 3

Next Sequence

not $111\ldots1$

Copy Sequence $2 \to 4$

$111\ldots1$

not all

Check Tape 4

all

Nondeterminism

- TMs
- PDAs
- FAs

NP completeness / complexity

- nondeterminism
- size of input

7.8. Universal Turing Machines

Definition 7.32. Universal Turing Machines

A *universal* Turing machine is a Turing machine T_u that works as follows. It is assumed to receive an input string of the form $e(T)e(z)$, where

- T is an arbitrary TM,
- z is a string over the input alphabet of T,
- and e is an encoding function whose values are strings in $\{0, 1\}^*$.

The computation performed by T_u on this input string satisfies these two properties:

1. T_u accepts the string $e(T)e(z)$ if and only if T accepts z.
2. If T accepts z and produces output y, then T_u produces output $e(y)$.

Some Crucial features of any encoding function e:

1. It should be possible to decide algorithmically, for any string $w \in \{0, 1\}^*$, whether w is a legitimate value of e.
2. A string w should represent at most one Turing machine, or at most one string z.
3. If $w = e(T)$ or $w = e(z)$, there should be an algorithm for *decoding* w.

Computability e itself. . .

Assumptions:

1. Names of the states are irrelevant.

2. Tape alphabet Γ of every Turing machine T is subset of infinite set $\mathcal{S} = \{a_1, a_2, a_3, \ldots\}$, where $a_1 = \Delta$.

Definition 7.33. An Encoding Function

Assign numbers to each state:
$n(h_a) = 1$, $n(h_r) = 2$, $n(q_0) = 3$, $n(q) \geq 4$ for other $q \in Q$.

Assign numbers to each tape symbol:
$n(a_i) = i$.

Assign numbers to each tape head direction:
$n(R) = 1$, $n(L) = 2$, $n(S) = 3$.

Definition 7.33. An Encoding Function (continued)

For each move m of T of the form $\delta(p,\sigma) = (q,\tau,D)$

$$e(m) = 1^{n(p)}01^{n(\sigma)}01^{n(q)}01^{n(\tau)}01^{n(D)}0$$

We list the moves of T in some order as $m_1, m_2, \ldots, m_k$, and we define

$$e(T) = e(m_1)0e(m_2)0\ldots0e(m_k)0$$

If $z = z_1z_2\ldots z_j$ is a string, where each $z_i \in \mathcal{S}$,

$$e(z) = 01^{n(z_1)}01^{n(z_2)}0\ldots01^{n(z_j)}0$$

Example 7.34. A Sample Encoding of a TM

Does $e(T)$ completely specify $T = (Q, \Sigma, \Gamma, q_0, \delta)$?

Definition 7.32. Universal Turing Machines

A *universal* Turing machine is a Turing machine T_u that works as follows. It is assumed to receive an input string of the form $e(T)e(z)$, where
- T is an arbitrary TM,
- z is a string over the input alphabet of T,
- and e is an encoding function whose values are strings in $\{0,1\}^*$.

The computation performed by T_u on this input string satisfies these two properties:
1. T_u accepts the string $e(T)e(z)$ if and only if T accepts z.
2. If T accepts z and produces output y, then T_u produces output $e(y)$.

Some Crucial features of any encoding function e:

1. It should be possible to decide algorithmically, for any string $w \in \{0,1\}^*$, whether w is a legitimate value of e.
2. A string w should represent at most one Turing machine with a given input alphabet Σ, or at most one string z.
3. If $w = e(T)$ or $w = e(z)$, there should be an algorithm for *decoding* w.

Computability e itself...

Huiswerkopgave 1

Inleveren: dinsdag 3 maart 2015, 13:45 uur