Fundamentele Informatica 3

voorjaar 2014

http://www.liacs.nl/home/rvvliet/fi3/

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

college 8, 31 maart 2014

8. Recursively Enumerable Languages
8.5. Not Every Language is Recursively Enumerable

9. Undecidable Problems
9.1. A Language That Can't Be Accepted,
and a Problem That Can't Be Decided

Huiswerkopgave 2,
inleverdatum 1 april 2014, 13:45 uur

A slide from lecture 7

Chomsky hierarchy

3	reg. languages	reg. grammar	FA	reg. expression
2	cf. languages	cf. grammar	PDA	
1	cs. languages	cs. grammar	LBA	
0	re. languages	unrestr. grammar	TM	

$$\mathcal{S}_3 \subseteq \mathcal{S}_2 \subseteq \mathcal{S}_1 \subseteq \mathcal{R} \subseteq \mathcal{S}_0$$

(modulo Λ)

8.5. Not Every Language is Recursively Enumerable

From Fundamentele Informatica 1:

Definition 8.23.
A Set A of the Same Size as B or Larger Than B

Two sets A and B, either finite or infinite, are the same size if there is a bijection $f : A \to B$.

A is larger than B if some subset of A is the same size as B but A itself is not.

From Fundamentele Informatica 1:

Definition 8.24.
Countably Infinite and Countable Sets

A set A is *countably infinite* (the same size as $\mathbb{N}$) if there is a bijection $f : \mathbb{N} \to A$, or a list $a_0, a_1, \ldots$ of elements of A such that every element of A appears exactly once in the list.

A is *countable* if A is either finite or countably infinite.

Theorem 8.25.
Every infinite set has a countably infinite subset, and every subset of a countable set is countable.

Proof. . .

(proof of second claim is Exercise 8.35. . .)

Example 8.26. The Set $\mathbb{N} \times \mathbb{N}$ is Countable

$$\mathbb{N} \times \mathbb{N} = \{(i, j) \mid i, j \in \mathbb{N}\}$$

although $\mathbb{N} \times \mathbb{N}$ looks much bigger than $\mathbb{N}$

$$\begin{array}{ccccc} (0,0) & (0,1) & (0,2) & (0,3) & \ldots \\ (1,0) & (1,1) & (1,2) & (1,3) & \ldots \\ (2,0) & (2,1) & (2,2) & (2,3) & \ldots \\ (3,0) & (3,1) & (3,2) & (3,3) & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{array}$$

Example 8.28.
A Countable Union of Countable Sets Is Countable

$$S = \bigcup_{i=0}^{\infty} S_i$$

Same construction as in Example 8.26, but. . .

Example 8.29. Languages Are Countable Sets

$$L \subseteq \Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$$

Two ways to list Σ^*

A slide from lecture 4

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.

A slide from lecture 4

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 $a_1 = \Delta$.

A slide from lecture 4

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:
$n(R) = 1$, $n(L) = 2$, $n(S) = 3$.

A slide from lecture 4

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 $z = z_1 z_2 \ldots z_j$ 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 8.30. The Set of Turing Machines Is Countable

Let $\mathcal{T}(\Sigma)$ be set of Turing machines with input alphabet Σ
There is injective function $e : \mathcal{T}(\Sigma) \to \{0,1\}^*$
(e is encoding function)

Hence (. . .), set of recursively enumerable languages is countable

Exercise 8.41.

For each case below, determine whether the given set is countable or uncountable. Prove your answer.

a. The set of all three-element subsets of $\mathbb{N}$.

b. The set of all finite subsets of $\mathbb{N}$.

Example 8.31. The Set $2^{\mathbb{N}}$ Is Uncountable

Hence, because $\mathbb{N}$ and $\{0, 1\}^*$ are the same size, there are uncountably many languages over $\{0, 1\}$

Example 8.31. The Set $2^{\mathbb{N}}$ Is Uncountable (continued)

No list of subsets of $\mathbb{N}$ is complete,
i.e., every list $A_0, A_1, A_2, \ldots$ of subsets of $\mathbb{N}$ leaves out at least one.

Take

$$A = \{i \in \mathbb{N} \mid i \notin A_i\}$$

Example 8.31. The Set $2^{\mathbb{N}}$ Is Uncountable (continued)

$$A = \{i \in \mathbb{N} \mid i \notin A_i\}$$

$$\begin{aligned}
A_0 &= \{0, 2, 5, 9, \ldots\} \\
A_1 &= \{1, 2, 3, 8, 12, \ldots\} \\
A_2 &= \{0, 3, 6\} \\
A_3 &= \emptyset \\
A_4 &= \{4\} \\
A_5 &= \{2, 3, 5, 7, 11, \ldots\} \\
A_6 &= \{8, 16, 24, \ldots\} \\
A_7 &= \mathbb{N} \\
A_8 &= \{1, 3, 5, 7, 9, \ldots\} \\
A_9 &= \{n \in \mathbb{N} \mid n > 12\} \\
&\ldots
\end{aligned}$$

	0	1	2	3	4	5	6	7	8	9	...
$A_0 = \{0, 2, 5, 9, \ldots\}$	1	0	1	0	0	1	0	0	0	1	...
$A_1 = \{1, 2, 3, 8, 12, \ldots\}$	0	1	1	1	0	0	0	0	1	0	...
$A_2 = \{0, 3, 6\}$	1	0	0	1	0	0	1	0	0	0	...
$A_3 = \emptyset$	0	0	0	0	0	0	0	0	0	0	...
$A_4 = \{4\}$	0	0	0	0	1	0	0	0	0	0	...
$A_5 = \{2, 3, 5, 7, 11, \ldots\}$	0	0	1	1	0	1	0	1	0	0	...
$A_6 = \{8, 16, 24, \ldots\}$	0	0	0	0	0	0	0	0	1	0	...
$A_7 = \mathbb{N}$	1	1	1	1	1	1	1	1	1	1	...
$A_8 = \{1, 3, 5, 7, 9, \ldots\}$	0	1	0	1	0	1	0	1	0	1	...
$A_9 = \{n \in \mathbb{N} \mid n > 12\}$	0	0	0	0	0	0	0	0	0	0	...
...						...					

	0	1	2	3	4	5	6	7	8	9	...
$A_0 = \{0, 2, 5, 9, \ldots\}$	1	0	1	0	0	1	0	0	0	1	...
$A_1 = \{1, 2, 3, 8, 12, \ldots\}$	0	1	1	1	0	0	0	0	1	0	...
$A_2 = \{0, 3, 6\}$	1	0	0	1	0	0	1	0	0	0	...
$A_3 = \emptyset$	0	0	0	0	0	0	0	0	0	0	...
$A_4 = \{4\}$	0	0	0	0	1	0	0	0	0	0	...
$A_5 = \{2, 3, 5, 7, 11, \ldots\}$	0	0	1	1	0	1	0	1	0	0	...
$A_6 = \{8, 16, 24, \ldots\}$	0	0	0	0	0	0	0	0	1	0	...
$A_7 = \mathbb{N}$	1	1	1	1	1	1	1	1	1	1	...
$A_8 = \{1, 3, 5, 7, 9, \ldots\}$	0	1	0	1	0	1	0	1	0	1	...
$A_9 = \{n \in \mathbb{N} \mid n > 12\}$	0	0	0	0	0	0	0	0	0	0	...
...						...					
$A = \{2, 3, 6, 8, 9, \ldots\}$	0	0	1	1	0	0	1	0	1	1	...

Hence, there are uncountably many subsets of $\mathbb{N}$.

Theorem 8.32. Not all languages are recursively enumerable. In fact, the set of languages over $\{0, 1\}$ that are not recursively enumerable is uncountable.

Proof. . .

(including Exercise 8.38)

Exercise 8.38.

Show that is S is uncountable and T is countable, then $S - T$ is uncountable.

Suggestion: proof by contradiction.

Theorem 8.25.
Every infinite set has a countably infinite subset, and every subset of a countable set is countable.

Proof. . .

(proof of second claim is Exercise 8.35. . .)

9. Undecidable Problems

9.1. A Language That Can't Be Accepted, and a Problem That Can't Be Decided

A slide from lecture 5:

Definition 8.1. Accepting a Language and Deciding a Language

A Turing machine T with input alphabet Σ accepts a language $L \subseteq \Sigma^*$,
if $L(T) = L$.

T *decides* L,
if T computes the characteristic function $\chi_L : \Sigma^* \to \{0, 1\}$

A language L is *recursively enumerable*,
if there is a TM that accepts L,

and L is *recursive*,
if there is a TM that decides L.

Example 8.31. The Set $2^{\mathbb{N}}$ Is Uncountable

Hence, because $\mathbb{N}$ and $\{0, 1\}^*$ are the same size, there are uncountably many languages over $\{0, 1\}$

Example 8.31. The Set $2^{\mathbb{N}}$ Is Uncountable (continued)

No list of subsets of $\mathbb{N}$ is complete,
i.e., every list $A_0, A_1, A_2, \ldots$ of subsets of $\mathbb{N}$ leaves out at least one.

Take

$$A = \{i \in \mathbb{N} \mid i \notin A_i\}$$

Example 8.31. The Set $2^{\mathbb{N}}$ Is Uncountable (continued)

$$A = \{i \in \mathbb{N} \mid i \notin A_i\}$$

$$\begin{aligned}
A_0 &= \{0, 2, 5, 9, \ldots\} \\
A_1 &= \{1, 2, 3, 8, 12, \ldots\} \\
A_2 &= \{0, 3, 6\} \\
A_3 &= \emptyset \\
A_4 &= \{4\} \\
A_5 &= \{2, 3, 5, 7, 11, \ldots\} \\
A_6 &= \{8, 16, 24, \ldots\} \\
A_7 &= \mathbb{N} \\
A_8 &= \{1, 3, 5, 7, 9, \ldots\} \\
A_9 &= \{n \in \mathbb{N} \mid n > 12\} \\
&\ldots
\end{aligned}$$

	0	1	2	3	4	5	6	7	8	9	...
$A_0 = \{0, 2, 5, 9, \ldots\}$	1	0	1	0	0	1	0	0	0	1	...
$A_1 = \{1, 2, 3, 8, 12, \ldots\}$	0	1	1	1	0	0	0	0	1	0	...
$A_2 = \{0, 3, 6\}$	1	0	0	1	0	0	1	0	0	0	...
$A_3 = \emptyset$	0	0	0	0	0	0	0	0	0	0	...
$A_4 = \{4\}$	0	0	0	0	1	0	0	0	0	0	...
$A_5 = \{2, 3, 5, 7, 11, \ldots\}$	0	0	1	1	0	1	0	1	0	0	...
$A_6 = \{8, 16, 24, \ldots\}$	0	0	0	0	0	0	0	0	1	0	...
$A_7 = \mathbb{N}$	1	1	1	1	1	1	1	1	1	1	...
$A_8 = \{1, 3, 5, 7, 9, \ldots\}$	0	1	0	1	0	1	0	1	0	1	...
$A_9 = \{n \in \mathbb{N} \mid n > 12\}$	0	0	0	0	0	0	0	0	0	0	...
...	...										

	0	1	2	3	4	5	6	7	8	9	...
$A_0 = \{0, 2, 5, 9, \ldots\}$	1	0	1	0	0	1	0	0	0	1	...
$A_1 = \{1, 2, 3, 8, 12, \ldots\}$	0	1	1	1	0	0	0	0	1	0	...
$A_2 = \{0, 3, 6\}$	1	0	0	1	0	0	1	0	0	0	...
$A_3 = \emptyset$	0	0	0	0	0	0	0	0	0	0	...
$A_4 = \{4\}$	0	0	0	0	1	0	0	0	0	0	...
$A_5 = \{2, 3, 5, 7, 11, \ldots\}$	0	0	1	1	0	1	0	1	0	0	...
$A_6 = \{8, 16, 24, \ldots\}$	0	0	0	0	0	0	0	0	1	0	...
$A_7 = \mathbb{N}$	1	1	1	1	1	1	1	1	1	1	...
$A_8 = \{1, 3, 5, 7, 9, \ldots\}$	0	1	0	1	0	1	0	1	0	1	...
$A_9 = \{n \in \mathbb{N} \mid n > 12\}$	0	0	0	0	0	0	0	0	0	0	...
...						...					
$A = \{2, 3, 6, 8, 9, \ldots\}$	0	0	1	1	0	0	1	0	1	1	...

Hence, there are uncountably many subsets of $\mathbb{N}$.

Set-up of Example 8.31:

1. Start with list of all subsets of $\mathbb{N}$: $A_0, A_1, A_2, \ldots$,
   each one associated with specific element of $\mathbb{N}$ (namely i)

2. Define another subset A by:
   $i \in A \iff i \notin A_i$

3. Conclusion: for all i, $A \neq A_i$
   Hence, contradiction
   Hence, there are uncountably many subsets of $\mathbb{N}$

Set-up of constructing language that is not RE:

1. Start with list of all RE languages over $\{0,1\}$ (which are subsets of $\{0,1\}^*$): $L(T_0), L(T_1), L(T_2), \ldots$ each one associated with specific element of $\{0,1\}^*$

2. Define another language L by: $x \in L \iff x \notin$ (language that x is associated with)

3. Conclusion: for all i, $L \neq L(T_i)$ Hence, L is not RE

	$e(T_0)$	$e(T_1)$	$e(T_2)$	$e(T_3)$	$e(T_4)$	$e(T_5)$	$e(T_6)$	$e(T_7)$	$e(T_8)$	$e(T_9)$
$L(T_0)$	1	0	1	0	0	1	0	0	0	1
$L(T_1)$	0	1	1	1	0	0	0	0	1	0
$L(T_2)$	1	0	0	1	0	0	1	0	0	0
$L(T_3)$	0	0	0	0	0	0	0	0	0	0
$L(T_4)$	0	0	0	0	1	0	0	0	0	0
$L(T_5)$	0	0	1	1	0	1	0	1	0	0
$L(T_6)$	0	0	0	0	0	0	0	0	1	0
$L(T_7)$	1	1	1	1	1	1	1	1	1	1
$L(T_8)$	0	1	0	1	0	1	0	1	0	1
$L(T_9)$	0	0	0	0	0	0	0	0	0	0
...						...				

	$e(T_0)$	$e(T_1)$	$e(T_2)$	$e(T_3)$	$e(T_4)$	$e(T_5)$	$e(T_6)$	$e(T_7)$	$e(T_8)$	$e(T_9)$
$L(T_0)$	1	0	1	0	0	1	0	0	0	1
$L(T_1)$	0	1	1	1	0	0	0	0	1	0
$L(T_2)$	1	0	0	1	0	0	1	0	0	0
$L(T_3)$	0	0	0	0	0	0	0	0	0	0
$L(T_4)$	0	0	0	0	1	0	0	0	0	0
$L(T_5)$	0	0	1	1	0	1	0	1	0	0
$L(T_6)$	0	0	0	0	0	0	0	0	1	0
$L(T_7)$	1	1	1	1	1	1	1	1	1	1
$L(T_8)$	0	1	0	1	0	1	0	1	0	1
$L(T_9)$	0	0	0	0	0	0	0	0	0	0
...						...				
NSA	0	0	1	1	0	0	1	0	1	1

Hence, *NSA* is not recursively enumerable.

A slide from lecture 4:

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.

Set-up of constructing language *NSA* that is not RE:

1. Start with list of all RE languages over $\{0, 1\}$ (which are subsets of $\{0, 1\}^*$): $L(T_0), L(T_1), L(T_2), \ldots$ each one associated with specific element of $\{0, 1\}^*$ (namely $e(T_i)$)

2. Define another language *NSA* by: $e(T_i) \in \textit{NSA} \iff e(T_i) \notin L(T_i)$

3. Conclusion: for all i, $\textit{NSA} \neq L(T_i)$ Hence, *NSA* is not RE

Set-up of constructing language *NSA* that is not RE:

1. Start with collection of all RE languages over $\{0, 1\}$ (which are subsets of $\{0, 1\}^*$): $\{L(T) \mid \text{ TM } T\}$ each one associated with specific element of $\{0, 1\}^*$ (namely $e(T)$)

2. Define another language *NSA* by: $e(T) \in \mathit{NSA} \iff e(T) \notin L(T)$

3. Conclusion: for all TM T, $\mathit{NSA} \neq L(T)$ Hence, *NSA* is not RE

Set-up of constructing language L that is not RE:

1. Start with list of all RE languages over $\{0,1\}$ (which are subsets of $\{0,1\}^*$): $L(T_0), L(T_1), L(T_2), \ldots$ each one associated with specific element of $\{0,1\}^*$ (namely x_i)

2. Define another language L by:
   $x_i \in L \iff x_i \notin L(T_i)$

3. Conclusion: for all i, $L \neq L(T_i)$
   Hence, L is not RE

Every infinite list $x_0, x_1, x_2, \ldots$ of different elements of $\{0,1\}^*$ yields language L that is not RE

Definition 9.1. The Languages *NSA* and *SA*

Let

$$\begin{aligned} \textit{NSA} &= \{e(T) \mid T \text{ is a TM, and } e(T) \notin L(T)\} \\ \textit{SA} &= \{e(T) \mid T \text{ is a TM, and } e(T) \in L(T)\} \end{aligned}$$

(*NSA* and *SA* are for “non-self-accepting” and “self-accepting.”)

A slide from lecture 4:

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.

Theorem 9.2. The language NSA is not recursively enumerable. The language SA is recursively enumerable but not recursive.

Proof. . .

Exercise 9.2.

Describe how a universal Turing machine could be used in the proof that SA is recursively enumerable.

Decision problem: problem for which the answer is ‘yes’ or ‘no’:

Given . . . , is it true that . . . ?

yes-instances of a decision problem:
instances for which the answer is ‘yes’

no-instances of a decision problem:
instances for which the answer is ‘no’

Decision problems

Given an undirected graph $G = (V, E)$,
does G contain a Hamiltonian path?

Given a list of integers $x_1, x_2, \ldots, x_n$,
is the list sorted?

Self-Accepting: Given a TM T, does T accept the string $e(T)$?

Three languages corresponding to this problem:

1. *SA*: strings representing yes-instances
2. *NSA*: strings representing no-instances
3. . . .

Self-Accepting: Given a TM T, does T accept the string $e(T)$?

Three languages corresponding to this problem:
1. *SA*: strings representing yes-instances
2. *NSA*: strings representing no-instances
3. E': strings not representing instances

For general decision problem P,
an encoding e of instances I as strings $e(I)$ over alphabet Σ
is called *reasonable*, if

1. there is algorithm to decide if string over Σ is encoding $e(I)$
2. e is injective
3. string $e(I)$ can be decoded

A slide from lecture 4:

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.

For general decision problem P and reasonable encoding e,

$$\begin{aligned} Y(P) &= \{e(I) \mid I \text{ is yes-instance of } P\} \\ N(P) &= \{e(I) \mid I \text{ is no-instance of } P\} \\ E(P) &= Y(P) \cup N(P) \end{aligned}$$

$E(P)$ must be recursive