Fundamentele Informatica 3

voorjaar 2016

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

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

college 10, 11 april 2016

9. Undecidable Problems
9.3. More Decision Problems Involving Turing Machines
9.4. Post's Correspondence Problem

A slide from lecture 9:

Definition 9.6. Reducing One Decision Problem to Another, and Reducing One Language to Another

Suppose P_1 and P_2 are decision problems. We say P_1 is reducible to P_2 ($P_1 \leq P_2$)

- if there is an algorithm
- that finds, for an arbitrary instance I of P_1, an instance $F(I)$ of P_2,
- such that

  for every I the answers for the two instances are the same,
  or I is a yes-instance of P_1
    if and only if $F(I)$ is a yes-instance of P_2.

A slide from lecture 9:

Theorem 9.7. Suppose $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$, and $L_1 \leq L_2$. If L_2 is recursive, then L_1 is recursive.

Suppose P_1 and P_2 are decision problems, and $P_1 \leq P_2$. If P_2 is decidable, then P_1 is decidable.

Proof. . .

A slide from lecture 9:

Theorem 9.8. Both *Accepts* and *Halts* are undecidable.

Proof.

1. Prove that *Self-Accepting* $\leq$ *Accepts* . . .

2. Prove that *Accepts* $\leq$ *Halts* . . .

9.3. More Decision Problems Involving Turing Machines

Accepts: Given a TM T and a string x, is $x \in L(T)$?
Instances are . . .

Halts: Given a TM T and a string x, does T halt on input x ?
Instances are . . .

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

Now fix a TM T:
T-*Accepts*: Given a string x, does T accept x ?
Instances are . . .
Decidable or undecidable ? (cf. **Exercise 9.7.**)

Exercise 9.7.

As discussed at the beginning of Section 9.3, there is at least one TM T such that the decision problem

“Given w, does T accept w ?”

is unsolvable.

Show that every TM accepting a nonrecursive language has this property.

Theorem 9.9. The following five decision problems are undecidable.

1. *Accepts*-Λ: Given a TM T, is $\Lambda \in L(T)$?

Proof.

1. Prove that *Accepts* $\leq$ *Accepts*-Λ . . .

Reduction from *Accepts* to *Accepts*-Λ.

Instance of *Accepts* is (T_1, x) for TM T_1 and string x.
Instance of *Accepts*-Λ is TM T_2.

$T_2 = F(T_1, x) =$

$$\textit{Write}(x) \rightarrow T_1$$

T_2 accepts Λ, if and only if T_1 accepts x.

If we had an algorithm/TM A_2 to solve *Accepts*-Λ,
then we would also have an algorithm/TM A_1 to solve *Accepts*,
as follows:

A_1:
Given instance (T_1, x) of *Accepts*,
1. construct $T_2 = F(T_1, x)$;
2. run A_2 on T_2.

A_1 answers 'yes' for (T_1, x),
if and only if A_2 answers 'yes' for T_2,
if and only T_2 accepts Λ,
if and only if T_1 accepts x.

Theorem 9.9. The following five decision problems are undecidable.

2. *AcceptsEverything*:
   Given a TM T with input alphabet Σ, is $L(T) = \Sigma^*$?

Proof.

2. Prove that *Accepts*-$\Lambda \leq$ *AcceptsEverything* . . .

Theorem 9.9. The following five decision problems are undecidable.

3. *Subset*: Given two TMs T_1 and T_2, is $L(T_1) \subseteq L(T_2)$?

Proof.

3. Prove that *AcceptsEverything* $\leq$ *Subset* . . .

Theorem 9.9. The following five decision problems are undecidable.

4. *Equivalent*: Given two TMs T_1 and T_2, is $L(T_1) = L(T_2)$

Proof.

4. Prove that *Subset* $\leq$ *Equivalent* . . .

Theorem 9.9. The following five decision problems are undecidable.

5. *WritesSymbol*:
   Given a TM T and a symbol a in the tape alphabet of T, does T ever write a if it starts with an empty tape ?

Proof.

5. Prove that *Accepts-Λ* $\leq$ *WritesSymbol* . . .

AtLeast10MovesOn-Λ:
Given a TM T, does T make at least ten moves on input Λ ?

WritesNonblank: Given a TM T, does T ever write a nonblank symbol on input Λ ?

Theorem 9.10.
The decision problem *WritesNonblank* is decidable.

Proof. . .

Definition 9.11. A Language Property of TMs

A property R of Turing machines is called a *language property* if, for every Turing machine T having property R, and every other TM T_1 with $L(T_1) = L(T)$, T_1 also has property R.

A language property of TMs is *nontrivial* if there is at least one TM that has the property and at least one that doesn't.

In fact, a language property is a property *of the languages accepted by* TMs.

Theorem 9.12. Rice's Theorem
If R is a nontrivial language property of TMs, then the decision problem

P_R: Given a TM T, does T have property R ?

is undecidable.

Proof. . .

Prove that *Accepts*-$\Lambda \leq P_R$ . . .

(or that *Accepts*-$\Lambda \leq P_{\text{not}-R}$ . . .)

T_2 highly unspecified. . .

A slide from lecture 9:

Definition 9.6. Reducing One Decision Problem to Another, and Reducing One Language to Another

Suppose P_1 and P_2 are decision problems. We say P_1 is reducible to P_2 ($P_1 \leq P_2$)

- if there is an algorithm
- that finds, for an arbitrary instance I of P_1, an instance $F(I)$ of P_2,
- such that

  for every I the answers for the two instances are the same,
  or I is a yes-instance of P_1
  if and only if $F(I)$ is a yes-instance of P_2.

Examples of decision problems to which Rice's theorem can be applied:

1. *Accepts-L*: Given a TM T, is $L(T) = L$? (assuming . . .)
2. *AcceptsSomething*:
   Given a TM T, is there at least one string in $L(T)$?
3. *AcceptsTwoOrMore*:
   Given a TM T, does $L(T)$ have at least two elements ?
4. *AcceptsFinite*: Given a TM T, is $L(T)$ finite ?
5. *AcceptsRecursive*:
   Given a TM T, is $L(T)$ recursive ? (note that . . .)

All these problems are undecidable.

Rice's theorem cannot be applied (directly)

- if the decision problem does not involve just one TM

Equivalent: Given two TMs T_1 and T_2, is $L(T_1) = L(T_2)$

Rice's theorem cannot be applied (directly)

• if the decision problem does not involve just one TM
Equivalent: Given two TMs T_1 and T_2, is $L(T_1) = L(T_2)$

• if the decision problem involves the *operation* of the TM
WritesSymbol: Given a TM T and a symbol a in the tape alphabet of T, does T ever write a if it starts with an empty tape ?
WritesNonblank: Given a TM T, does T ever write a nonblank symbol on input Λ ?

• if the decision problem involves a *trivial* property
Accepts-NSA: Given a TM T, is $L(T) =$ *NSA* ?

9.4. Post's Correspondence Problem

Instance:

10	01	0	100	1
101	100	10	0	010

Instance:

10	01	0	100	1
101	100	10	0	010

Match:

10	1	01	0	100	100	0	100
101	010	100	10	0	0	10	0

Definition 9.14. Post's Correspondence Problem

An instance of Post's correspondence problem (*PCP*) is a set

$$\{(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_n, \beta_n)\}$$

of pairs, where $n \geq 1$ and the α_i's and β_i's are all nonnull strings over an alphabet Σ.

The decision problem is this:

Given an instance of this type, do there exist a positive integer k and a sequence of integers $i_1, i_2, \ldots, i_k$, with each i_j satisfying $1 \leq i_j \leq n$, satisfying

$$\alpha_{i_1}\alpha_{i_2}\ldots\alpha_{i_k} = \beta_{i_1}\beta_{i_2}\ldots\beta_{i_k} \quad ?$$

$i_1, i_2, \ldots, i_k$ need not all be distinct.

Definition 9.14. Post's Correspondence Problem (continued)

An instance of the modified Post's correspondence problem ($MPCP$) looks exactly like an instance of PCP, but now the sequence of integers is required to start with 1. The question can be formulated this way:

Do there exist a positive integer k and a sequence $i_2, i_3, \ldots, i_k$ such that

$$\alpha_1 \alpha_{i_2} \ldots \alpha_{i_k} = \beta_1 \beta_{i_2} \ldots \beta_{i_k} \quad ?$$

(Modified) correspondence system, match.

Theorem 9.15. *MPCP* $\leq$ *PCP*

Proof.

For instance

$$I = \{(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_n, \beta_n)\}$$

of *MPCP*, construct instance $J = F(I)$ of *PCP*, such that I is yes-instance, if and only if J is yes-instance.

For $1 \leq i \leq n$, if

$$(\alpha_i, \beta_i) = (a_1 a_2 \ldots a_r,\ b_1 b_2 \ldots b_s)$$

we let

$$(\alpha_i', \beta_i') = (a_1 \# a_2 \# \ldots a_r \#,\ \# b_1 \# b_2 \ldots \# b_s)$$

10	01	0	100	1
101	100	10	0	010

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

Match MPCP:

10	1	01	0	100	100	0	100
101	010	100	10	0	0	10	0

Almost match PCP:

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

Match MPCP:

10	1	01	0	100	100	0	100
101	010	100	10	0	0	10	0

Almost match PCP:

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

Match MPCP:

10	1	01	0	100	100	0	100
101	010	100	10	0	0	10	0

Almost match PCP:

1#	0#1#	0#	1#0#0#	1#0#0#	0#	1#0#0#	$
#0#1#0	#1#0#0	#1#0	#0	#0	#1#0	#0	#$

Match MPCP:

10	1	01	0	100	100	0	100
101	010	100	10	0	0	10	0

Almost match PCP:

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

Match MPCP:

10	1	01	0	100	100	0	100
101	010	100	10	0	0	10	0

Match PCP:

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

For $1 \leq i \leq n$, if

$$(\alpha_i, \beta_i) = (a_1 a_2 \dots a_r,\ b_1 b_2 \dots b_s)$$

we let

$$(\alpha_i', \beta_i') = (a_1 \# a_2 \# \dots a_r \#,\ \# b_1 \# b_2 \dots \# b_s)$$

If

$$(\alpha_1, \beta_1) = (a_1 a_2 \dots a_r,\ b_1 b_2 \dots b_s)$$

add

$$(\alpha_1'', \beta_1'') = (\# a_1 \# a_2 \# \dots a_r \#,\ \# b_1 \# b_2 \dots \# b_s)$$

Finally, add

$$(\alpha_{n+1}', \beta_{n+1}') = (\$, \#\$)$$

#1#0#
#1#0#1

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

$
#$