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 9, 7 april 2014

9. Undecidable Problems
9.1. A Language That Can't Be Accepted, and a Problem That Can't Be Decided
9.2. Reductions and the Halting Problem
9.3. More Decision Problems Involving Turing Machines

A slide from lecture 8:

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

Let

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

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

A slide from lecture 8:

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

Proof. . .

A slide from lecture 8:

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’

A slide from lecture 8:

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

A slide from lecture 8:

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 8:

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

Definition 9.3. Decidable Problems

If P is a decision problem, and e is a reasonable encoding of instances of P over the alphabet Σ, we say that P is *decidable* if $Y(P) = \{e(I) \mid\ I \text{ is a yes-instance of } P\}$ is a recursive language.

Theorem 9.4. The decision problem *Self-Accepting* is undecidable.

Proof. . .

For every decision problem, there is *complementary* problem P', obtained by changing ‘true’ to ‘false’ in statement.

Non-Self-Accepting:
Given a TM T, does T fail to accept $e(T)$?

Theorem 9.5. For every decision problem P, P is decidable if and only if the complementary problem P' is decidable.

Proof. . .

9.2. Reductions and the Halting Problem

(Informal) Examples of reductions

1. Recursive algorithms

2. Given NFA M and string x, is $x \in L(M)$?

3. Given FAs M_1 and M_2, is $L(M_1) \subseteq L(M_2)$?

Theorem 2.15.
Suppose $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$ are finite automata accepting L_1 and L_2, respectively.
Let M be the FA $(Q, \Sigma, q_0, A, \delta)$, where

$Q = Q_1 \times Q_2$

$q_0 = (q_1, q_2)$

and the transition function δ is defined by the formula

$\delta((p, q), \sigma) = (\delta_1(p, \sigma), \delta_2(q, \sigma))$

for every $p \in Q_1$, every $q \in Q_2$, and every $\sigma \in \Sigma$.

Then

1. If $A = \{(p, q) | \ p \in A_1$ or $q \in A_2\}$,
   M accepts the language $L_1 \cup L_2$.
2. If $A = \{(p, q) | \ p \in A_1$ and $q \in A_2\}$,
   M accepts the language $L_1 \cap L_2$.
3. If $A = \{(p, q) | \ p \in A_1$ and $q \notin A_2\}$,
   M accepts the language $L_1 - L_2$.

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.

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

If L_1 and L_2 are languages over alphabets Σ_1 and Σ_2, respectively, we say L_1 is reducible to L_2 ($L_1 \leq L_2$)

- if there is a Turing-computable function
- $f : \Sigma_1^* \to \Sigma_2^*$
- such that for every $x \in \Sigma_1^*$,

$$x \in L_1 \text{ if and only if } f(x) \in L_2$$

Less / more formal definitions.

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. . .

In context of decidability: decision problem $P \approx$ language $Y(P)$

Question

“is instance I of P a yes-instance ?”

is essentially the same as

“does string x represent yes-instance of P ?”,

i.e.,

“is string $x \in Y(P)$?”

Therefore, $P_1 \leq P_2$, if and only if $Y(P_1) \leq Y(P_2)$.

Two more decision problems:

Accepts: Given a TM T and a string w, is $w \in L(T)$?

Halts: Given a TM T and a string w, does T halt on input w ?

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

Proof.

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

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

Proof.

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

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

Application:

```
n = 4;
while (n is the sum of two primes)
  n = n+2;
```

This program loops forever, if and only if Goldbach's conjecture is true.

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*-$\Lambda \leq$ *WritesSymbol* . . .