Fundamentele Informatica 3

voorjaar 2012

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

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

college 15, maandag 14 mei 2012
laatste hoorcollege

9. Undecidable Problems
9.4. Post's Correspondence Problem
9.5. Undecidable Problems
Involving Context-Free Languages

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.

(similar for languages)

Theorem 9.7.

(statement about languages)

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.

Two more decision problems:

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

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

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

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...

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

. . .

2. *AcceptsSomething*:

   Given a TM T, is there at least one string in $L(T)$?

. . .

All these problems are undecidable.

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 $i_2, i_3, \ldots, k$ 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 $1 \le i \le n$, if

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

we let

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

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

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

If

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

add

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

Finally, add

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

Theorem 9.16. *Accepts* $\leq$ *MPCP*

The technical details of the proof of this result do not have to be known for the exam. However, one must be able to carry out the construction below.

Proof. . .

For every instance (T, w) of *Accepts*, construct instance $F(T, w)$ of *MPCP*, such that . . .

Notation:

description of tape contents: $x\underline{\sigma}y$ or $x\underline{y}$

configuration $xqy = xqy\Delta = xqy\Delta\Delta$

initial configuration corresponding to input x: $q_0\Delta x$

In the third edition of the book, a configuration is denoted as $(q, x\underline{y})$ or $(q, x\underline{\sigma}y)$ instead of xqy or $xq\sigma y$.
This old notation is also allowed for Fundamentele Informatica 3.

Proof of Theorem 9.16. (continued)

Take
$(\alpha_1, \beta_1) = (\#, \#q_0\Delta w\#)$

Pairs of type 1: (a, a) for every $a \in \Gamma \cup \{\Delta\}$, and $(\#, \#)$

Pairs of type 2: corresponding to moves in T, e.g.,
(qa, bp), if $\delta(q, a) = (p, b, R)$
(cqa, pcb), if $\delta(q, a) = (p, b, L)$
$(q\#, pa\#)$, if $\delta(q, \Delta) = (p, b, S)$

Pairs of type 3: for every $a, b \in \Gamma \cup \{\Delta\}$, the pairs
$(h_a a, h_a)$, $(a h_a, h_a)$, $(a h_a b, h_a)$

One pair of type 4:
$(h_a\#\#, \#)$

Proof of Theorem 9.16. (continued)

Two assumptions in book:

1. T never moves to h_r
2. $w \neq \Lambda$ (i.e., special initial pair if $w = \Lambda$)

These assumptions are not necessary...

Theorem 9.17.
Post's correspondence problem is undecidable.

Example 9.18. A Modified Correspondence System for a TM

T accepts all strings in $\{a, b\}^*$ ending with b.

Example 9.18. A Modified Correspondence System for a TM (continued)

$$
\begin{array}{llll}
(q_0\Delta, \Delta q_1) & (q_0\#, \Delta q_1\#) & (q_1 a, a q_1) & (q_1 b, b q_1) \\
(a q_1 \Delta, q_2 a \Delta) & (b q_1 \Delta, q_2 b \Delta) & \ldots &
\end{array}
$$

9.5. Undecidable Problems Involving Context-Free Languages

For an instance

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

of *PCP*, let. . .

CFG G_α be defined by productions

$$S_\alpha \to \alpha_i S_\alpha c_i \mid \alpha_i c_i \quad (1 \leq i \leq n)$$

CFG G_β be defined by productions

$$S_\beta \to \beta_i S_\beta c_i \mid \beta_i c_i \quad (1 \leq i \leq n)$$

Theorem 9.20.
These two problems are undecidable:

1. *CFGNonEmptyIntersection*:
   Given two CFGs G_1 and G_2, is $L(G_1) \cap L(G_2)$ nonempty?

2. *IsAmbiguous*:
   Given a CFG G, is G ambiguous?

Proof. . .

Let T be TM, let x be string accepted by T, and let

$$z_0 \vdash z_1 \vdash z_2 \vdash z_3 \ldots \vdash z_n$$

be ‘succesful computation’ of T for x,
i.e., $z_0 = q_0 \Delta x$
and z_n is accepting configuration.

Let T be TM, let x be string accepted by T, and let

$$z_0 \vdash z_1 \vdash z_2 \vdash z_3 \ldots \vdash z_n$$

be ‘succesful computation’ of T for x,
i.e., $z_0 = q_0\Delta x$
and z_n is accepting configuration.

Successive configurations z_i and z_{i+1} are almost identical;
hence $z_i\#z_{i+1}$ cannot be described by CFG,
cf. $XX = \{xx \mid x \in \{a,b\}^*\}$.

$z_i\#z_{i+1}^r$ is almost a palindrome, and *can* be described by CFG.

Definition 9.21. Valid Computations of a TM

Let $T = (Q, \Sigma, \Gamma, q_0, \delta)$ be a Turing machine.

A *valid computation* of T is a string of the form

$$z_0 \# z_1^r \# z_2 \# z_3^r \ldots \# z_n \#$$

if n is even, or

$$z_0 \# z_1^r \# z_2 \# z_3^r \ldots \# z_n^r \#$$

if n is odd,
where in either case, $\#$ is a symbol not in Γ,
and the strings z_i represent successive configurations of T on soms input string x, starting with the initial configuration z_0 and ending with an accepting configuration.

The set of valid computations of T will be denoted by C_T.

Part of Theorem 9.22. For a TM T, the complement C'_T of C_T is a context-free language.

In fact C'_T can be described as the union of seven context-free languages, for each of which we can algorithmically construct a CFG.

The proof of this result does not have to be known for the exam.

Theorem 9.23. The decision problem

> *CFGGeneratesAll*: Given a CFG G with terminal alphabet Σ, is $L(G) = \Sigma^*$?

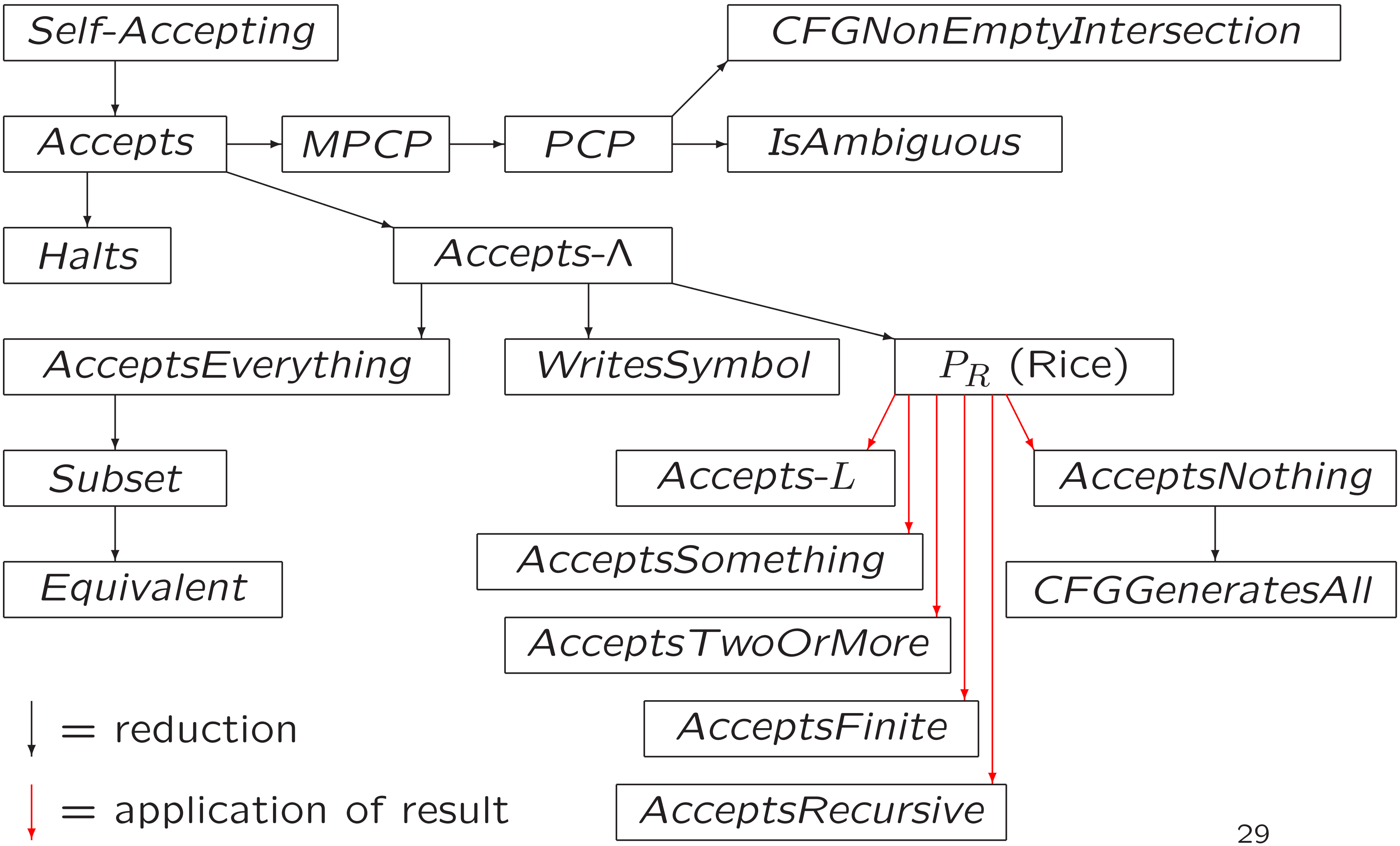
is undecidable.

Proof.

Let
AcceptsNothing: Given a TM T, is $L(T) = \emptyset$?

Prove that *AcceptsNothing* $\leq$ *CFGGeneratesAll* . . .

Undecidable Decision Problems (we have discussed)



Tentamen: maandag 11 juni 2012, 10:00–13:00

Vragenuur. . . ?

Volgend jaar: hoofdstuk 7–10 ipv hoofdstuk 5–9.