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 11, 21 april 2015

9. Undecidable Problems
9.5. Undecidable Problems
Involving Context-Free Languages
10. Computable Functions
10.1. Primitive Recursive Functions

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 10

9.4. Post's Correspondence Problem

Instance:

10	01	0	100	1
101	100	10	0	010

A slide from lecture 10

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

A slide from lecture 10

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.

A slide from lecture 10

Theorem 9.17.
Post's correspondence problem is undecidable.

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...

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

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

Example derivation:

$S_\alpha \Rightarrow \alpha_2 S_\alpha c_2 \Rightarrow \alpha_2 \alpha_5 S_\alpha c_5 c_2 \Rightarrow \alpha_2 \alpha_5 \alpha_1 S_\alpha c_1 c_5 c_2 \Rightarrow \alpha_2 \alpha_5 \alpha_1 \alpha_3 c_3 c_1 c_5 c_2$

Unambiguous

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

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

Example.

Let I be the following instance of PCP:

10	01	0	100	1
101	100	10	0	010

G_α and G_β. . .

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. . .

Theorem 9.20.
This problem is undecidable:

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

Alternative proof. . .

Let CFG G_1 be defined by productions

$$S_1 \to \alpha_i S_1 \beta_i^r \ \mid \ \alpha_i \# \beta_i^r \quad (1 \le i \le n)$$

Let CFG G_2 be defined by productions

$$S_2 \to a S_2 a \ \mid \ b S_2 b \ \mid \ a \# a \ \mid \ b \# b$$

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 the language

$$\{z\#z'\# \mid \; z \text{ and } z' \text{ are config's of } T \text{ for which } z \vdash z'\}$$

cannot be described by CFG,
cf. $XX = \{xx \mid x \in \{a,b\}^*\}$.

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.

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

Lemma.

The language

$$L_1 \;=\; \{z\#(z')^r\# \mid \; z \text{ and } z' \text{ are config's of } T \text{ for which } z \vdash z'\}$$

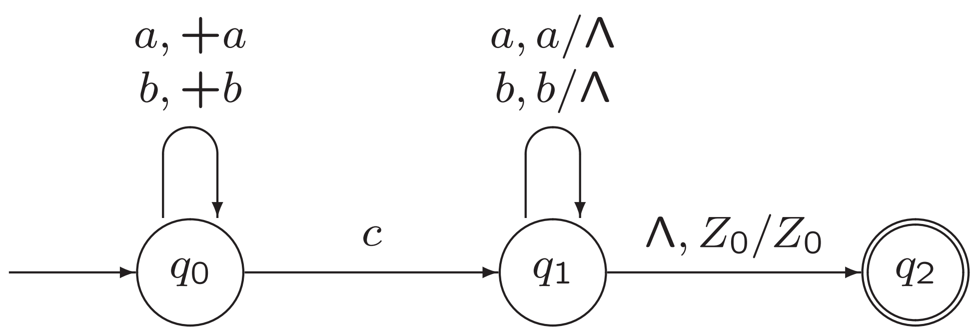
is context-free.

Proof. . .

A slide from lecture 1

Example 5.3. A Pushdown Automaton Accepting *SimplePal*

SimplePal $= \{xcx^r \mid x \in \{a,b\}^*\}$



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 some 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.

Theorem 9.22.

For a TM $T = (Q, \Sigma, \Gamma, q_0, \delta)$,
• the set C_T of valid computations of T is the intersection of two context-free languages,
• and its complement C'_T is a context-free language.

Proof. . .

Theorem 9.22.

For a TM $T = (Q, \Sigma, \Gamma, q_0, \delta)$,
• the set C_T of valid computations of T is the intersection of two context-free languages,
• and its complement C'_T is a context-free language.

Proof. Let

$$\begin{aligned}
L_1 &= \{z\#(z')^r\# \mid \ z \text{ and } z' \text{ are config's of } T \text{ for which } z \vdash z'\} \\
L_2 &= \{z^r\#z'\# \mid \ z \text{ and } z' \text{ are config's of } T \text{ for which } z \vdash z'\} \\
I &= \{z\# \mid \ z \text{ is initial configuration of } T\} \\
A &= \{z\# \mid \ z \text{ is accepting configuration of } T\} \\
A_1 &= \{z^r\# \mid \ z \text{ is accepting configuration of } T\}
\end{aligned}$$

$$C_T = L_3 \cap L_4$$

where

$$\begin{aligned} L_3 &= I L_2^*(A_1 \cup \{\Lambda\}) \\ L_4 &= L_1^*(A \cup \{\Lambda\}) \end{aligned}$$

for each of which we can algorithmically construct a CFG

If $x \in C'_T$ (i.e., $x \notin C_T$), then. . .

If $x \in C'_T$ (i.e., $x \notin C_T$), then

1. Either, x does not end with #

Otherwise, let $x = z_0\#z_1\#\ldots\#z_k\#$

(no reversed strings in this partitioning)

2. Or, for some even i, z_i is not configuration of T

3. Or, for some odd i, z_i^r is not configuration of T

4. Or z_0 is not initial configuration of T

5. Or z_k is neither accepting configuration, nor the reverse of one

6. Or, for some even i, $z_i \not\vdash z_{i+1}^r$

7. Or, for some odd i, $z_i^r \not\vdash z_{i+1}$

If $x \in C'_T$ (i.e., $x \notin C_T$), then

1. Either, x does not end with $\#$

Otherwise, let $x = z_0 \# z_1 \# \ldots \# z_k \#$

2. Or, for some even i, z_i is not configuration of T
3. Or, for some odd i, z_i^r is not configuration of T
4. Or z_0 is not initial configuration of T
5. Or z_k is neither accepting configuration, nor the reverse of one
6. Or, for some even i, $z_i \not\vdash z_{i+1}^r$
7. Or, for some odd i, $z_i^r \not\vdash z_{i+1}$

Hence, C'_T is union of seven context-free languages,
for each of which we can algorithmically construct a CFG

Corollary.

The decision problem

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

is undecidable (cf. Theorem 9.20(1)).

Proof.

Let
AcceptsSomething: Given a TM T, is $L(T) \neq \emptyset$?

Prove that *AcceptsSomething* $\leq$ *CFGNonEmptyIntersection*

Study this result yourself.

Theorem 9.23. The decision problem

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

is undecidable.

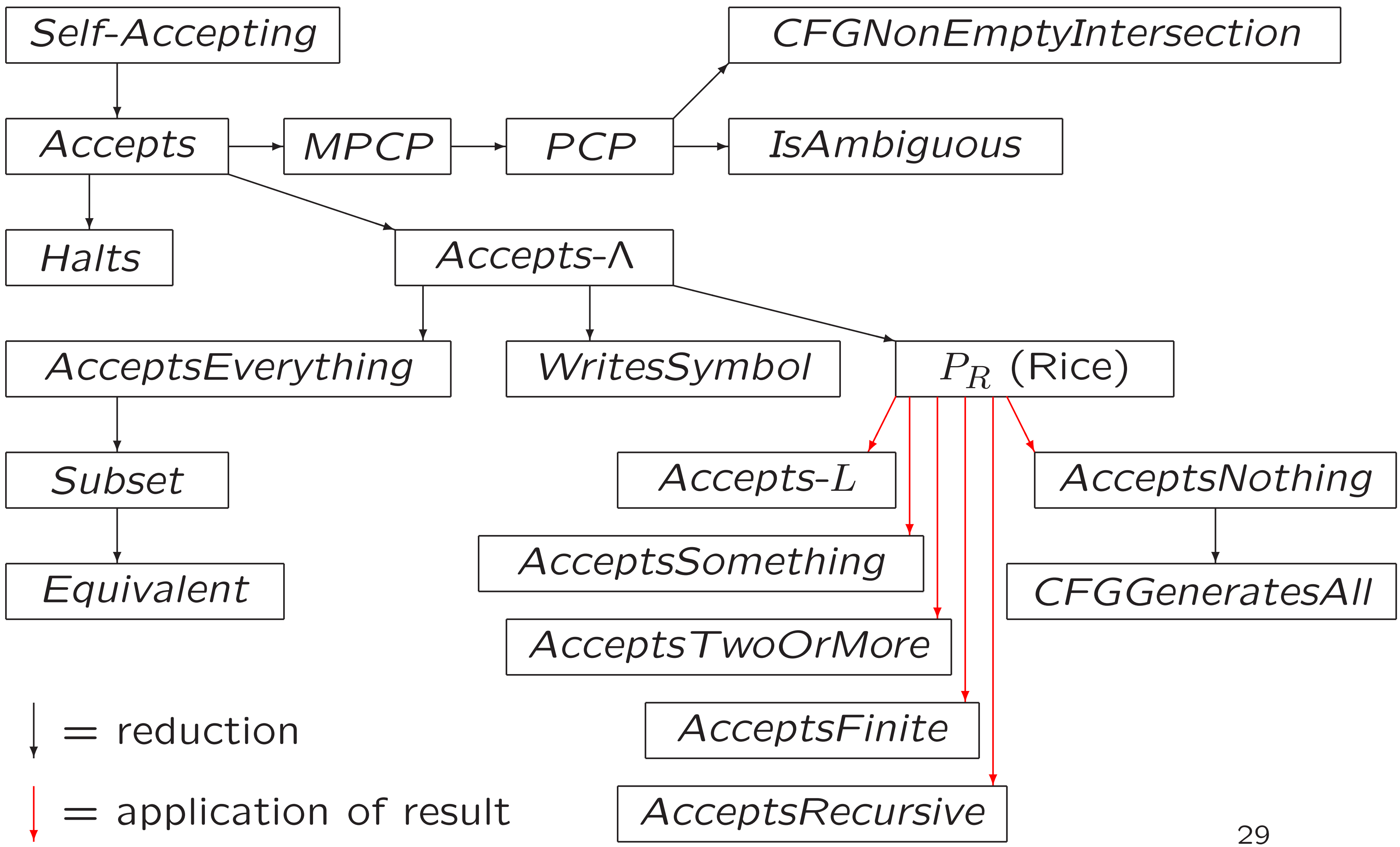
Proof.

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

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

Study this result yourself.

Undecidable Decision Problems (we have discussed)



10. Computable Functions

10.1. Primitive Recursive Functions

Definition 10.1. Initial Functions

The initial functions are the following:

1. *Constant* functions: For each $k \geq 0$ and each $a \geq 0$, the constant function $C_a^k : \mathbb{N}^k \to \mathbb{N}$ is defined by the formula

$$C_a^k(X) = a \quad \text{for every } X \in \mathbb{N}^k$$

Definition 10.1. Initial Functions

The initial functions are the following:

1. *Constant* functions: For each $k \geq 0$ and each $a \geq 0$, the constant function $C_a^k : \mathbb{N}^k \to \mathbb{N}$ is defined by the formula

$$C_a^k(X) = a \quad \text{for every } X \in \mathbb{N}^k$$

2. The *successor* function $s : \mathbb{N} \to \mathbb{N}$ is defined by the formula

$$s(x) = x + 1$$

Definition 10.1. Initial Functions

The initial functions are the following:

1. *Constant* functions: For each $k \geq 0$ and each $a \geq 0$, the constant function $C_a^k : \mathbb{N}^k \to \mathbb{N}$ is defined by the formula

$$C_a^k(X) = a \quad \text{for every } X \in \mathbb{N}^k$$

2. The *successor* function $s : \mathbb{N} \to \mathbb{N}$ is defined by the formula

$$s(x) = x + 1$$

3. *Projection* functions: For each $k \geq 1$ and each i with $1 \leq i \leq k$, the projection function $p_i^k : \mathbb{N}^k \to \mathbb{N}$ is defined by the formula

$$p_i^k(x_1, x_2, \ldots, x_k) = x_i$$

Definition 10.2. The Operations of Composition and Primitive Recursion

1. Suppose f is a partial function from $\mathbb{N}^k$ to $\mathbb{N}$, and for each i with $1 \leq i \leq k$, g_i is a partial function from $\mathbb{N}^m$ to $\mathbb{N}$.
   The partial function obtained from f and $g_1, g_2, \ldots, g_k$ by composition is the partial function h from $\mathbb{N}^m$ to $\mathbb{N}$ defined by the formula

   $$h(X) = f(g_1(X), g_2(X), \ldots, g_k(X)) \text{ for every } X \in \mathbb{N}^m$$

Definition 10.2. The Operations of Composition and Primitive Recursion (continued)

2. Suppose $n \geq 0$ and g and h are functions of n and $n+2$ variables, respectively. (By "a function of 0 variables," we mean simply a constant.)
   The function obtained from g and h by the operation of *primitive recursion* is the function $f : \mathbb{N}^{n+1} \to \mathbb{N}$ defined by the formulas

$$\begin{aligned} f(X, 0) &= g(X) \\ f(X, k+1) &= h(X, k, f(X, k)) \end{aligned}$$

   for every $X \in \mathbb{N}^n$ and every $k \geq 0$.

Example 10.5. Addition, Multiplication and Subtraction

$$Add(x, y) = x + y$$

Definition 10.3. Primitive Recursive Functions

The set *PR* of *primitive recursive* functions is defined as follows.

1. All initial functions are elements of *PR*.

2. For every $k \geq 0$ and $m \geq 0$, if $f : \mathbb{N}^k \to \mathbb{N}$ and $g_1, g_2, \ldots, g_k : \mathbb{N}^m \to \mathbb{N}$ are elements of *PR*, then the function $f(g_1, g_2, \ldots, g_k)$ obtained from f and $g_1, g_2, \ldots, g_k$ by composition is an element of *PR*.

3. For every $n \geq 0$, every function $g : \mathbb{N}^n \to \mathbb{N}$ in *PR*, and every function $h : \mathbb{N}^{n+2} \to \mathbb{N}$ in *PR*, the function $f : \mathbb{N}^{n+1} \to \mathbb{N}$ obtained from g and h by primitive recursion is in *PR*.

In other words, the set *PR* is the smallest set of functions that contains all the initial functions and is closed under the operations of composition and primitive recursion.

Example 10.5. Addition, Multiplication and Subtraction

$$\mathit{Mult}(x, y) = x * y$$

Example 10.5. Addition, Multiplication and Subtraction

$$\mathit{Sub}(x, y) = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$$

$x \mathbin{\dot{-}} y$