Es)

$\Sigma = \{a, b, c\}$

$L_1 = \{ w \mid x u y v z ,\ x, y, z \in \Sigma ,\ u, v \in \Sigma^* ,$
$x = z ,\ |u| = |v| \}$

$xyz = ab^n c b^n a$
$\overset{!}{=} \underbrace{ab \dots b}_{xy} c b^n a$

if $y = a$, $xy^0z = b^n c b^n a \notin L_1$
if $y = ab^*$, $xy^0z = b^{n-|y|-1} c b^n a \notin L_1$
if $y = b^*$, $xy^0z = ab^{n-|y|} c b^n a$
↳ if $|y|$ is odd $\rightarrow xy^0z \notin L_1$
if $|y|$ is even $\rightarrow xy^0z \in L_1$!

↳ Can't use the Pumping lemma

$R_0 = (a+b+c)$

$R_1 = R_0R_0R_0(R_0R_0)^*$ → odd number of symbols

$R_2 = aR_0^*a + bR_0^*b + cR_0^*c$ → first = last

$R_1 \cap R_2$ generates L_1

or

$$R = \underbrace{a\overbrace{R_0(R_0R_0)^*}^{\text{odd length}}a}_{|w| \geq 3} + bR_0(R_0R_0)^*b + cR_0(R_0R_0)^*c$$

first = last

$L_1 \in REG$

$L_2 = \{ w \mid w = x u y v z,\ x, y, z \in \Sigma,\ u, v \in \Sigma^*,\ x = y = z \}$

$R_0 = a + b + c$

$R = a R_0^* a R_0^* a + b R_0^* b R_0^* b + c R_0^* c R_0^* c$

$L_2 \in REG$

$L_3 = \{ w \mid w \in x u y v z,\ x, y, z \in \Sigma,\ u, v \in \Sigma^*,\ x = y = z \}$

$xyz = ab^n a b^n a$

$\overset{!}{=} \underbrace{ab \ldots b}_{xy} b a b^n a$

if $y = a$, $xy^0 z = b^n a b^n a \notin L_3$

if $y = b^+$, $xy^0 z = a b^{\underline{n - |y|}} a b^{\underline{n}} a \notin L_3$

if $y = ab^+$, $xy^0 z = b^{\underline{n - |y|} - 1} a b^{\underline{n}} a \notin L_3$

Automata, Languages and Computation

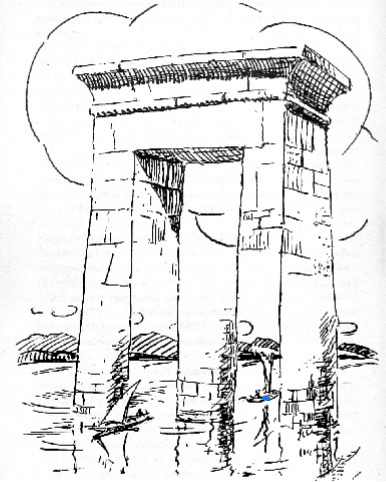
MOST DIFFICULT !!!

Chapter 9 : Undecidability

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Lecture based on material originally developed by :
Gösta Grahne, Concordia University

Undecidability



Recursively enumerable languages

From now onward : Modern computers = Turing machines

A language L is **recursively enumerable** (RE) if $L = L(M)$ for some TM M

Given an input string w, M halts if $w \in L(M)$, but M **may not halt** if $w \notin L(M)$

Recursive languages

A language L is **recursive** (REC) or, equivalently, the decision problem L represents is **decidable**, if $L = L(M)$ for a TM M that halts for **every** input

A recursive/decidable language corresponds to the definition of **algorithm**, for which we impose that computation halts both for positive and negative instances of the problem

String indexing

Let us sort all strings in $\{0, 1\}^*$:

- by length
- lexicografically, for strings of the same length

i	string
1	ϵ
2	0
3	1
4	00
5	01
$\vdots$	$\vdots$

We associate with each string a positive integer i called **index**

String indexing

We write w_i to denote the i-th string

We can easily verify that, for each $w \in \{0, 1\}^*$, we have

$$w = w_i \quad \Leftrightarrow \quad i = 1w$$

Encoding of TM

We now want to encode a TM **with binary input alphabet** $M = (Q, \{0, 1\}, \Gamma, \delta, q_1, B, F\}$ by means of a binary string, which we denote $\mathrm{enc}(M)$

We need to assign integers to each state, tape symbol, and symbols L and R indicating directions

We rename the states as $q_1, q_2, \ldots, q_r$. Initial state: q_1, final state: q_2 (unique)

We rename the tape symbols as $X_1, X_2, \ldots, X_s$. Also: $0 = X_1$, $1 = X_2$, $B = X_3$

$L = D_1$ and $R = D_2$

Encoding of TM

For the transition function, if

$$\delta(q_i, X_j) = (q_k, X_l, D_m)$$

the binary code C for the transition is (we use unary notation for i, j, k, l, m)

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

Note : We never have two consecutive occurrences of 1, since $i, j, k, l, m \geqslant 1$ is always satisfied

Encoding of TM

For a TM, we concatenate the codes C_i for all transitions, separated by 11

$$C_1 11 C_2 11 \cdots 11 C_{n-1} 11 C_n$$

no specific order

There are several codes for M, obtained by indexing the symbols and/or listing the transitions in different orders

Many binary strings do not correspond to a TM
Example : 11001 or 001110

Note : In the following we write enc(M) to denote a generic code for M; keep in mind that enc() **is not** a function.

Try to draw a map between set of all TMs and set of binary strings, representing the encoding relation

Example

Let $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$, where δ is defined as

D_2 0100100010100

$$\delta(q_1, 1) = (q_3, 0, R) \quad \delta(q_3, 0) = (q_1, 1, R)$$
$$\delta(q_3, 1) = (q_2, 0, R) \quad \delta(q_3, B) = (q_3, 1, L)$$

Transition encodings C_i

0100100010100 0001010100100
00010010010100 0001000100010010

TM encoding enc(M)

0100100010100**11**0001010100100**11**00010010010100**11**0001000100010010

TM indexing

1234567890

We can now **enumerate** all TM (with repetition) using positive integers as indices and using our string indexing

For $i \geqslant 1$, the i-th TM M_i is defined as follows

- if w_i is a valid encoding representing TM M, then $M_i = M$
- if w_i is not a valid encoding, then M_i is the TM that halts immediately for any input (only one state and no transition, $L(M_i) = \emptyset$)

Diagonalization language

The following table reports whether M_i accepts (1) or rejects (0) w_j

ALL RE languages

All turing machines

$j \longrightarrow$ +∞

$i \downarrow$	1	2	3	4	...
1	0	1	1	0	...
2	1	1	0	0	...
3	0	0	1	1	...
4	0	1	0	1	...
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

+∞

Diagonal

{ tutte righe con solo ∅...

010101010 è la prima TM che non dà empty