Ecole Polytechnique de Louvain

LINMA2111 Algorithmes et complexité
CM 10

Mélanie Sedda - Benoît Sluysmans

30 mai 2015

Halting problem :
Input : A Matlab code "M.m", an input for this matlab code
Output : YES if M(x) halts after a finite time, NO otherwise

This is a decision problem, i.e. a problem with output YES or NO. A decision problem is decidable iff it is computable.

Theorem (Turing;1936) : The halting problem for Matlab machines is undecidable.
Proof : By contradiction : we assume that there is a Matlab code "Halt.m" solving the halting problem. We build "Diagonal.m", a program wich take as argument another Matlab code "M.m" with a string of char as input.

Algorithm 1 Diagonal(M)

```
if halt(M,M) = YES % M halts on input "M.m" then
  while TRUE % infinite loop
else
  STOP % if M does not halt on M, then stop
end if
```

What happens if we call Diagonal(Diagonal) ?
– If it stops then halt(Diagonal,Diagonal) = YES $\Rightarrow$ infinite loop, does not stop
– If it does not stop, then it stops

$\Rightarrow$ Contradiction
$\Rightarrow$ "Halt.m" does not exist.

$\square$

This is called a diagonal argument. Why ?

We create this array :

	x_1	x_2	x_3	x_4	...
M1.m	**Halt**	NotHalt	NotHalt	Halt	...
M2.m	Halt	**Halt**	NotHalt	Halt	...
M3.m	NotHalt	NotHalt	**NotHalt**	Halt	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Where x_i represent all strings of char, and "M.i" represent all codes with string of char as inputs. We have entry(i,j) = Halt iff Mi(x_j) halts.

We now change all entries of the diagonal :
If Mi(x_i) = "Halt" then Diagonal(i) = "NotHalt" and conversely.

In our case, we obtain (Diagonal(1),Diagonal(2),Diagonal(3),...) = ("NotHalt","NotHalt","Halt",...). This row is not in the array, by construction. That means that "Diagonal.m"$\neq$"Mi.m, $\forall i$.

Cantor (1891) invented the diagonal argument to prove that $[0,1]$ (or $\mathbb{R}$, etc) is not countable, i.e. it does not exist a surjection $\mathbb{N} \to [0,1]$. If there exists a surjection $c : \mathbb{N} \to [0,1]$, it follows that $[0,1] = \{c_0, c_1, c_2, ...\}$.

We can now create the array

c_0	0.	**1**	0	2	...
c_1	0.	3	**9**	1	...
c_2	0.	0	0	**1**	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

From which we create a new number $x = 0.202...$ $(0 \to 1, 1 \to 2, 2 \to 3, ..., 9 \to 0)$. We now see that x is not in the array, by construction $\Rightarrow x \neq c_i \ \forall i$, which is a contradiction.

$\square$

What about the decision problems (from integer or string of char to YES or NO) ?

a	b	...	aa	...
0	1	2	3	...
YES	YES	NO	YES	...
1	1	0	1	...

Where the third row characterizes the decision proble P, and we can create $x_P \in [0,1]$ from the last row : $x_P = 0.1101...$. That means that $\forall x \in [0,1]$, I can create a decision problem. So decision problems are uncountable because in bijection with $[0,1]$.

Matlab codes are countable (e.g. string of char, ASCII codes $\Rightarrow$ integer $\in \mathbb{N}$). And every Matlab code can solve at most one decision problem. So most decision problems are undecidable ! But Turing's proof is interesting because it provides a specific, interesting decision problem that is undecidable.