19th April, 2019

→ Recall $\{0,1\}^\infty$: infinite binary strings is uncountable.

→ $\{0,1\}^*$: set of all finite binary strings

or

Σ^* (where Σ is a finite set of alphabets)

is countable.

→ Any SJava program is an element of Σ^*.

Proving Undecidability through reductions

Problem X. Prove X is undecidable.

(i) Start from scratch.

<u>Def^n:</u> We $A \leq B$ if given a program that decides B, we can use it as a subroutine to construct a program that decides A.

<u>Implication:</u> If A is a known undecidable problem, and $A \leq B$, then B must be undecidable.

* Acceptance problem.

$$L_{Accept} = \{ \langle M, x \rangle : M \text{ accepts the input } x \}$$

Thm L_{Accept} is undecidable.

Pf: $L_{Halt} \leq L_{Accept}$.

Assume acceptChecker decides L_{Accept}. Use this to build program haltchecker which decides L_{Halt}.

haltchecker on input $\langle M, x \rangle$ does the following:

* modify the code of M to M' such that M' accepts whenever M halts (irrespective of Ms output).

* return acceptChecker(M', x).

S Java skeleton code for haltChecker:

```
bool haltChecker ( string M , string x){
    M' = modify( M );
    return acceptChecker( M', x );
}

// code acceptChecker.

string Modify( string M ) {
    // modifies M to M'
    // s.t M' corresponds to
    // program described below.
}
```

Code for M'. // main is the first function.

```
bool main( string w ) {
```

```
        a = interpretr( M, w);
        return true;
}
```

// code for the Universal (Java program.

∎.

Zero-checker

$L_{Zero} = \{ M : M \text{ accepts input } 0 \}$.

__Thm__ L_{Zero} is undecidable.

__Pf:__ $L_{Halt} \leq L_{Zero}$.

Assume zeroChecker decides L_{Zero}.

haltchecker on input $\langle M, x \rangle$:

(i) Construct program M' which on input w behaves as follows:

(a) If $w \neq 0$, output false.

(b) $w = 0$: uses the universal SJava program to simulate M on x;
if M produces an output, ignore it, and return true.

(ii) return zeroChecker(M');

* If M halts on x then M' accepts the input 0.

* If M does not halt on x, then M' does not halt on input 0.

✓ $\square$.

→ SJava skeleton code for haltChecker

```
bool    haltChecker ( string M,  string x){
        M' = zerofocus ( M, x );
```

```
    return  zeroChecker(M');
}

// code for zeroChecker.

String  Zerofocus( string M,  sting n){
        // returns  source of program
        // M'  which  correspond to
        // program  described  below.
}
```

```
bool  main ( string w)  {
    if (w=0)  {
        a =  interpreter(M,n);
        return  true;
    }
    return false;
}
```

code for M'

// code for sJava simulator.

$\square$

Rice's Theorem.

L : subset of all sJava programs.

membership of program M in L is based on a 'semantic property' of M.

$L_{0,100} = \{ M :$ M accepts input 0 in at most 100 execution steps $\}$.

$\uparrow$
decidable!

[informal]
Rice's Theorem: All nontrivial L (of the above form) are undecidable.

$\square$