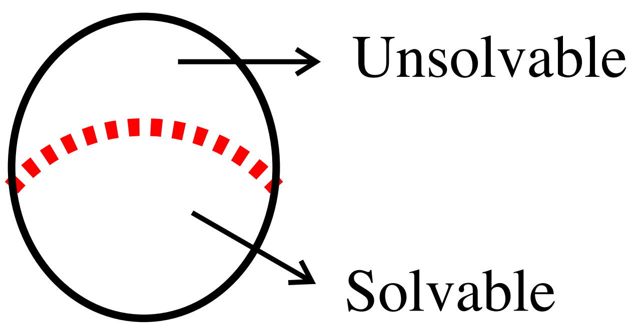
Review

Objective

techniques — how to prove that a problem is unsolvable

insights — what sort of problems are unsolvable



unsolvable (by algorithms) problems	$\rightsquigarrow$	undecidable languages
solvable problems	$\rightsquigarrow$	decidable languages

We say that Turing machine M **decides language L** if (and only if) M computes the function

$$f : \Sigma^* \to \{Y, N\} \text{ and } \begin{array}{l} \text{for each } x \in L : f(x) = Y \\ \text{for each } x \notin L : f(x) = N \end{array}$$

Language L is **(Turing) decidable** if (and only if) there is a Turing machine which decides it.

We say that Turing machine M **accepts language L** if M halts if and only if its input is an string in L.

Language L is **(Turing) acceptable** if (and only if) there is a Turing machine which accepts it.

Meaning

- All algorithms in the world live in the basket
- Infinitely many of them — most of them are unknown to us
- Meaning of unsolvability: No algorithm in the basket solves the problem (decides L)
- Meaning of solvability: There is an algorithm in the basket that solves the problem (but we don't necessarily know what the algorithm looks like)

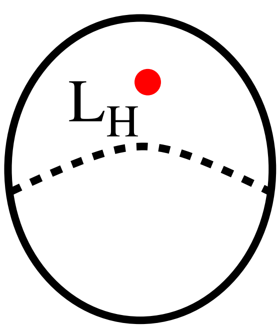
Techniques

To prove that

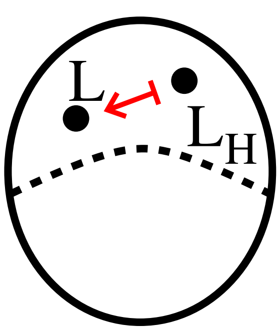
- **L is solvable:** Show an algorithm
- **L is unsolvable:** Difficulty: Cannot check all the algorithms in the basket. Cannot even see most of them, because they have not yet been constructed …

Strategy

1. Show L_H (HALTING problem) undecidable using diagonalisation .



2. Show another langauge L undecidable by **reduction**: If L kan be solved, so can L_H.



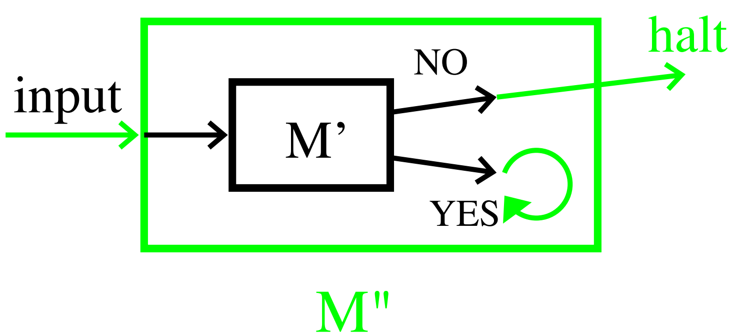
Step 1: HALTING is unsolvable

Def. 1 (HALTING)

$$L_H = \{(M, x) | M \textit{ halts on input } x\}$$

Theorem 1 *The Halting Problem is undecidable.*

Proof (by **diagonalization**): Given a Turing machine M' that decides L_H we can construct a Turing machine M'' as follows:

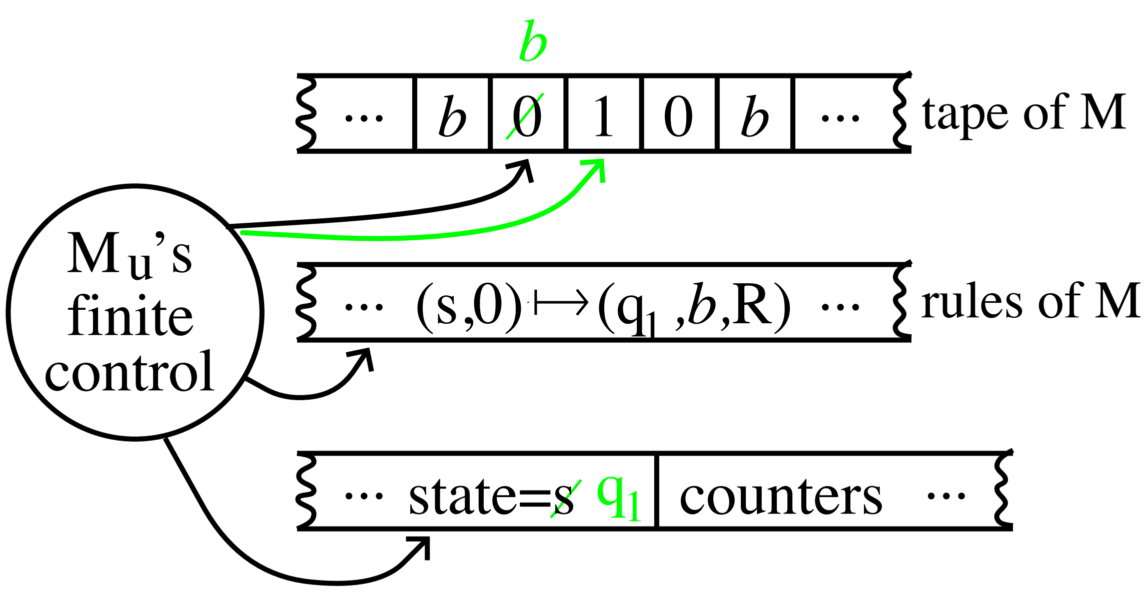


QUESTION: What does M'' do when given M'', M'' as input?

CONCLUSION: Since the assumption that M' exists leads to a contradiction (i.e. an impossible machine), it must be false.

The universal Turing machine M_u

- M_u works like an ordinary computer: It takes a code (program) M and a string x as input and simulates (runs) M on input x.
- M_u exists by Church's thesis.
- To **prove** existence of M_u we must construct it. Here is a 3-tape M_u:



Alternative proof of Theorem 1:

	ϵ	0	1	00	01	10	11	000	$\cdots$
ϵ									
0									
1									
00	1	0	0	$\cancel{1}0$	0	1	0	0	
01	0	1	0	1	$\cancel{0}1$	1	1	0	
10									
11									
000									
$\vdots$									

- We have strings as column labels
- We have Turing machine (codes) as row labels
- The 1's in each row define the set of strings each TM accepts.
- After flipping the diagonal elements, the 1's on the diagonal represents those machines which don't accept their own code as input
- No Turing machine can possibly accept that diagonal language!

Meaning

An example with $\Pi = 3.14159265359\ldots$:

$$L_1 = \{X | X \text{ is a substring of the decimal expansion of } \Pi\}$$

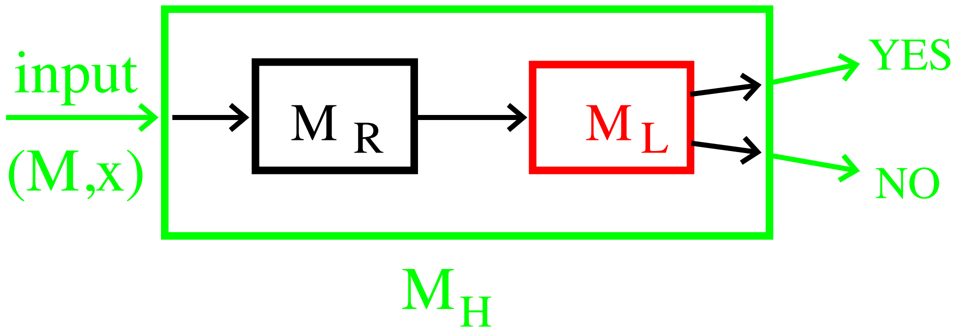
$$L_2 = \{K | \text{ There are } K \text{ consecutive zeros in the decimal expansion of } \Pi\}$$

Classify L_1, L_2 as

- not acceptable
- acceptable but not decidable
- decidable

Note: Only problems which take an infinite number of different inputs can possibly be unsolvable.

Reductions



Meaning of a reduction

Image: You meet an old friend with a brand new M_L-machine under his shoulder. Without even looking at the machine you say: “It is fake!”

How the reduction goes

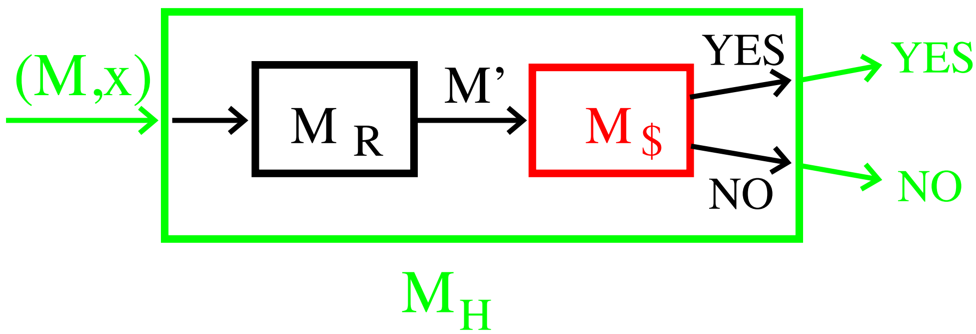
Image (an old riddle): You are standing at a crossroad deep in the forest. One way leads to the hungry crocodiles, the other way to the castle with the huge piles of gold. In front of you stands one of the two twin brothers. One of them always lies, the other always tells the truth. You can ask one question. What do you say?

A typical reduction

$$L_\$ = \{M | M \text{ (eventually) writes a } \$ \text{ when started with a blank tape}\}$$

Claim: $L_\$$ is undecidable

Proof:



M':

```
Simulate M on input x;
IF M halts THEN write a $;
```

Important points:

- M' must not write a \$ during the simulation of M!
- 'Write a \$' is an arbitrarly chosen action!

$\mathbf{M_R}$:

Output the M_u code modified as follows: Instead of reading its input M and x, the modified M_u has them stored in its finite control and it **writes them** on its tape. After that the modified M_u proceeds as the ordinary M_u untill the simulation is finished. Then it writes a \$.

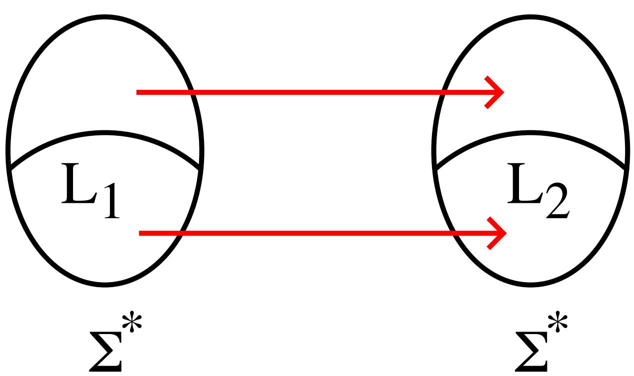
Reduction as mathematical function

Given a reduction from L_1 to L_2. Then M_R computes a function

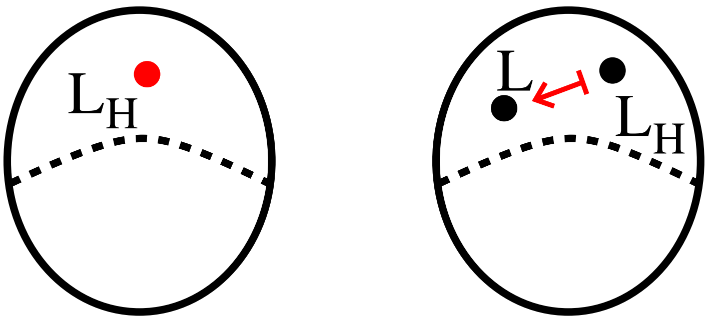
$$f_R : \sum^* \to \sum^*$$

which is such that

$$x \in L_1 \Rightarrow f_R(x) \in L_2$$
$$x \notin L_1 \Rightarrow f_R(x) \notin L_2$$

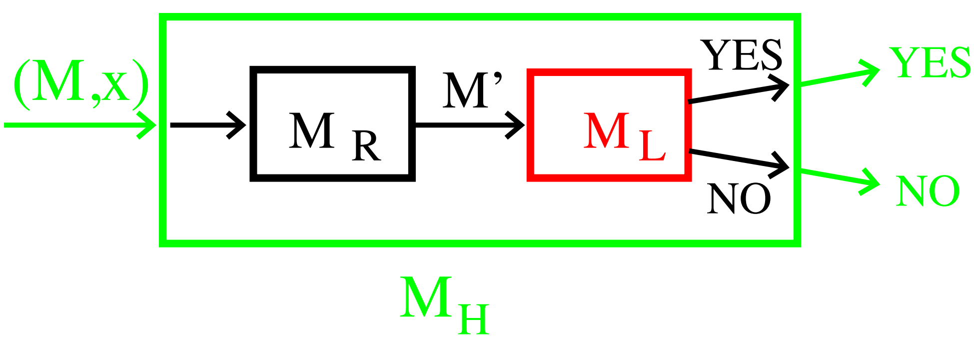


Undecidability in a Nutshell



- show L_H unsolvable by **diagonalization**
- show L unsolvable by **reduction**

Reductions



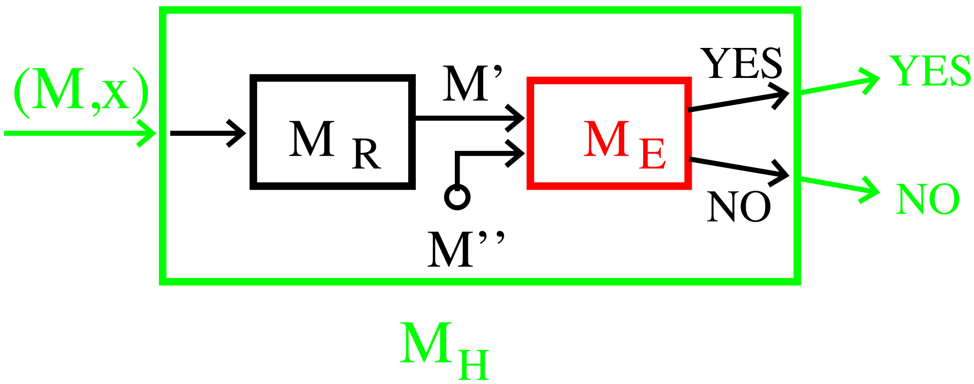
M’:

```
Simulate M on input x;
Do <ACTION>;
```

Example

Theorem 2 *Equivalence of programs (Turing machines) is undecidable.*

Proof:



M':

```
Simulate M on input x;
Accept;
```

M":

```
Accept;
```

- M'' accepts all inputs.
- M and x are constants to M'.
- M' accepts all inputs if and only if M halts on input x.

A solvable problem

$L_s = \{M_s | M_s$(eventually) moves its R/W head
when started with a blank tape$\}$

"Proof" that L_s is undecidable:

```
Simulate M on input x;
Move the R/W head;
```

"Proof" that L_s is decidable:

Simulate M_s on empty string as input;
for $|\Gamma| \times |Q|$ steps;