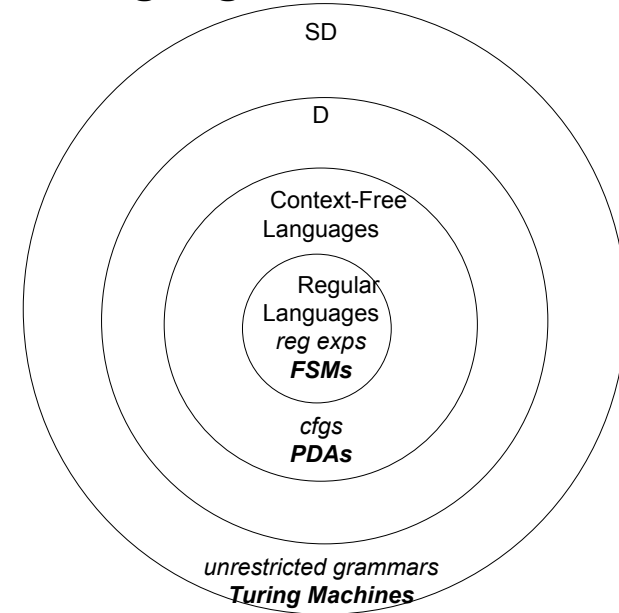


# The Unsolvability of the Halting Problem

## Chapter 19

## Languages and Machines



## D and SD

- A TM  $M$  with input alphabet  $\Sigma$  **decides** a language  $L \subseteq \Sigma^*$  iff, for any string  $w \in \Sigma^*$ ,
  - if  $w \in L$  then  $M$  accepts  $w$ , and
  - if  $w \notin L$  then  $M$  rejects  $w$ .

A language  $L$  is **decidable** (in D) iff there is a Turing machine  $M$  that decides it.

- A TM  $M$  with input alphabet  $\Sigma$  **semidecides**  $L$  iff for any string  $w \in \Sigma^*$ ,
  - if  $w \in L$  then  $M$  accepts  $w$
  - if  $w \notin L$  then  $M$  does not accept  $w$ .  $M$  may reject or loop.

A language  $L$  is **semidecidable** (in SD) iff there is a Turing machine that semidecides it.

## Defining the Universe

What is the complement of:

- $A^n B^n = \{a^n b^n : n \geq 0\}$

- $\{ \langle M, w \rangle : \text{TM } M \text{ halts on input string } w \}$ .

## Defining the Universe

$L_1 = \{ \langle M, w \rangle : \text{TM } M \text{ halts on input string } w \}.$

$L_2 = \{ \langle M \rangle : M \text{ halts on nothing} \}.$

$L_3 = \{ \langle M_a, M_b \rangle : M_a \text{ and } M_b \text{ halt on the same strings} \}.$

For a string  $w$  to be in  $L_1$ , it must:

- be syntactically well-formed.
- encode a machine  $M$  and a string  $w$  such that  $M$  halts when started on  $w$ .

Define the universe from which we are drawing strings to contain **only those strings that meet the syntactic requirements of the language definition.**

This convention has no impact on the decidability of any of these languages since the set of syntactically valid strings is in D.

## A Different Definition of Complement

Our earlier definition:

$$\neg L_1 = \{x: x \text{ is not a syntactically well formed } \langle M, w \rangle \text{ pair} \} \\ \cup \\ \{ \langle M, w \rangle : \text{TM } M \text{ does not halt on input string } w \}.$$

We will use a different definition:

Define the **complement** of any language  $L$  whose member strings include at least one Turing machine description to be with respect to a universe of strings that are of the same syntactic form as  $L$ .

Now we have:

$$\neg L_1 = \{ \langle M, w \rangle : \text{TM } M \text{ does not halt on input string } w \}.$$

## The Halting Language H

$H = \{ \langle M, w \rangle : \text{TM } M \text{ halts on input string } w \}$

**Theorem:** The language:

$H = \{ \langle M, w \rangle : \text{TM } M \text{ halts on input string } w \}$

- is **semidecidable**, but
- is **not decidable**.

## Does This Program Halt?

1. Concatenate 0 to the end of input
2. Halt.

## Does This Program Halt?

1. Concatenate 0 to the end of input
2. Move right
3. Go to 1.

## Does This Program Halt?

*times3*(*x*: positive integer) =

While  $x \neq 1$  do:

    If  $x$  is even then  $x = x/2$ .

    Else  $x = 3x + 1$

25	29	34	40	8
76	88	17	20	4
38	44	52	10	2
19	22	26	5	1
58	11	13	16	

<http://www.numbertheory.org/php/collatz.html>

## H is Semidecidable

**Lemma:** The language:

$H = \{ \langle M, w \rangle : \text{TM } M \text{ halts on input string } w \}$

is semidecidable.

**Proof:** The TM  $M_H$  semidecides H:

$M_H(\langle M, w \rangle) =$   
1. Run  $M$  on  $w$ .

$M_H$  halts iff  $M$  halts on  $w$ . Thus  $M_H$  semidecides H.

## The Unsolvability of the Halting Problem

**Lemma:** The language:

$H = \{ \langle M, w \rangle : \text{TM } M \text{ halts on input string } w \}$

is not decidable.

**Proof:** If H were decidable, then some TM  $M_H$  would decide it.  $M_H$  would implement the specification:

**halts**( $\langle M: \text{string}, w: \text{string} \rangle$ ) =

    If  $\langle M \rangle$  is a Turing machine description

    and  $M$  halts on input  $w$

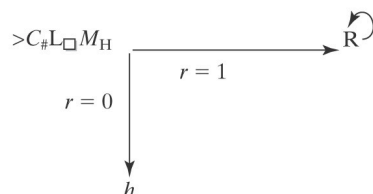
    then accept.

    Else reject.

## Trouble

**Trouble**( $x$ : string) =  
if **halts**( $x$ ,  $x$ ) then loop forever, else halt.

If there exists an  $M_H$  that computes the function **halts**, then put  $r=1$  if  $M_H$  halts and  $r=0$  otherwise. **Trouble** is then:



What is **Trouble**(**<Trouble>**)?

What is **halts**(**<Trouble, Trouble>**)?

- If **halts** reports that **Trouble**(**<Trouble>**) halts, **Trouble** loops.
- But if **halts** reports that **Trouble**(**<Trouble>**) does not halt, then **Trouble** halts.

## Viewing the Halting Problem as Diagonalization

- Lexicographically enumerate Turing machines:  $TM_i$ ,  $i = 1, 2, 3, \dots$
- Lexicographically enumerate all possible inputs:  $i_j$ ,  $j = 1, 2, 3, \dots$
- Build the table  $table[i, j] = 1$  if  $TM_i$  halts on  $j$  and 0 otherwise
- $M_H$  can be used to fill in any cell in the table

	$i_1$	$i_2$	$i_3$	...	<b>&lt;Trouble&gt;</b>	...
$TM_1$	1					
$TM_2$		1				
$TM_3$					1	
...				1		
<b>Trouble</b>			1			1
...	1	1	1			
...				1		

If **table**[**Trouble, Trouble**] = 1, then  $M_H$  says that **Trouble** halts on **Trouble**, so **Trouble** loops, a contradiction.

If **table**[**Trouble, Trouble**] = 0, then  $M_H$  says that **Trouble** does not halt on **Trouble**, so **Trouble** halts, a contradiction.

## If H were in D

$H = \{ \langle M \rangle, w : \text{TM } M \text{ halts on input string } w \}$

**Theorem:** If  $H$  were in  $D$  then every SD language would be in  $D$ .

**Proof:** Let  $L$  be any SD language. There exists a TM  $M_L$  that semidecides it.

If  $H$  were also in  $D$ , then there would exist an  $O$  that decides it.

## If H were in D

To decide whether  $w$  is in  $L(M_L)$ :

$M'(w: \text{string}) =$

1. Run  $O$  on  $\langle M_L, w \rangle$ .
2. If  $O$  accepts (i.e.,  $M_L$  will halt), then:
  - 2.1. Run  $M_L$  on  $w$ .
  - 2.2. If it accepts, accept. Else reject.
3. Else reject.

So, if  $H$  were in  $D$ , all SD languages would be.

# Back to the Entscheidungsproblem

**Theorem:** The Entscheidungsproblem is unsolvable.

**Proof:** (Due to Turing)

1. If we could solve the problem of determining whether a given Turing machine ever prints the symbol 0, then we could solve the problem of determining whether a given Turing machine halts.
2. But we can't solve the problem of determining whether a given Turing machine halts, so neither can we solve the problem of determining whether it ever prints 0.
3. Given a Turing machine  $M$ , we can construct a logical formula  $F$  that is true iff  $M$  ever prints the symbol 0.
4. If there were a solution to the Entscheidungsproblem, then we would be able to determine the truth of any logical sentence, including  $F$  and thus be able to decide whether  $M$  ever prints the symbol 0.
5. But we know that there is no procedure for determining whether  $M$  ever prints 0.
6. So there is no solution to the Entscheidungsproblem.

# Language Summary

