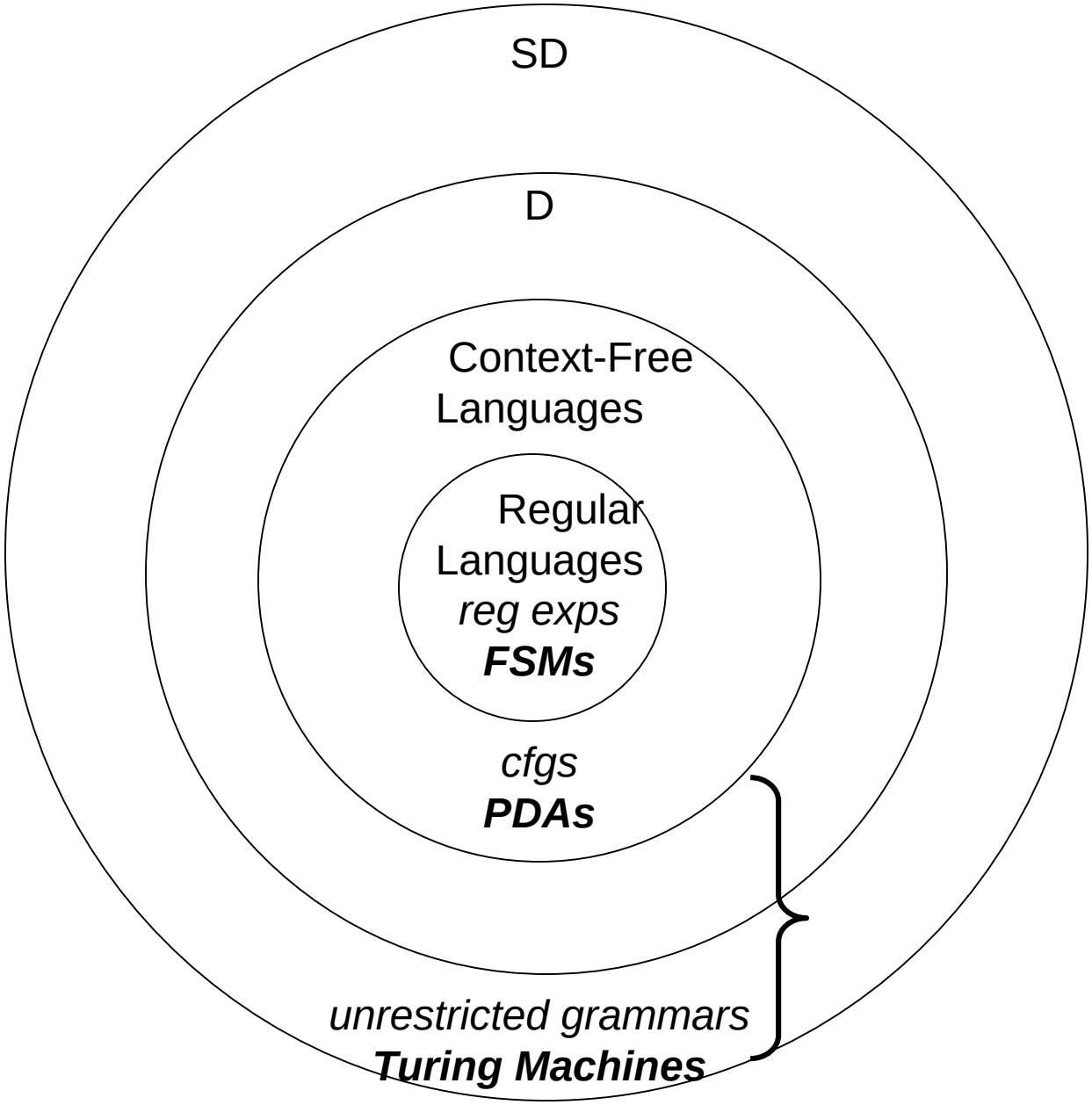
The Unsolvability of the Halting Problem

Chapter 19

Languages and Machines



D and SD

- A TM M with input alphabet Σ ***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 Σ ***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:

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

- {<*M*, *w*> : TM *M* halts on input string *w*}.

Defining the Universe

L_1 = {<*M*, *w*> : TM *M* halts on input string *w*}.
L_2 = {<*M*> : *M* halts on nothing}.
L_3 = {<M_a, M_b> : M_a and M_b 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* is not a syntactically well formed <*M*, *w*> pair}

$\cup$

{<*M*, *w*> : TM *M* 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:

$\neg L_1$ = {<*M*, *w*> : TM *M* does not halt on input string *w*}.

The Language H

H = {<*M*, *w*> : TM *M* halts on input string *w*}

Theorem: The language:

H = {<*M*, *w*> : TM *M* halts on input string *w*}

- is semidecidable, but
- is not decidable.

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

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
76
38
19
58

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
76	88
38	44
19	22
58	11

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
76	88	17
38	44	52
19	22	26
58	11	13

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
76	88	17	20
38	44	52	10
19	22	26	5
58	11	13	16

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 = {<*M*, *w*> : TM *M* halts on input string *w*}

is semidecidable.

Proof:

H is Semidecidable

Lemma: The language:

H = {<*M*>, *w* : TM *M* halts on input string *w*}

is semidecidable.

Proof: The TM M_H semidecides H:

M_H(<*M*, *w*>) =
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 = {<*M*, *w*> : TM *M* halts on input string *w*}

is not decidable.

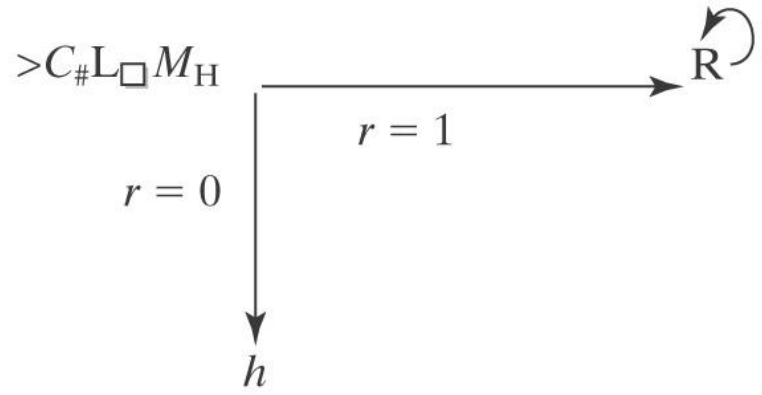
Proof: If H were decidable, then some TM M_H would decide it. M_H would implement the specification:

halts(<*M*: string, *w*: string>) =
    If <*M*> 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*, *Trouble* exists:



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.
- Let 1 mean halting, blank mean non halting.

	i_1	i_2	i_3	…	<*Trouble*>	…
$machine_1$	1					
$machine_2$		1				
$machine_3$					1	
…				1		
Trouble			1			1
…	1	1	1			
…				1		

But *Trouble* behaves as:

Trouble			1		1	1

Or maybe *halts* said that *trouble*(<*trouble*>) would halt. But then *trouble* would loop.

If H were in D

H = {<*M*>, *w* : TM *M* 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*: string) =
1. Run *O* on $<M_L, w>$.
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

IN		OUT
Semideciding TM	**SD** H	
Deciding TM	**D** $A^nB^nC^n$	Diagonalize
CF grammar PDA Closure	**Context-Free** A^nB^n	Pumping Closure
Regular Expression FSM	**Regular** a*b*	Pumping Closure