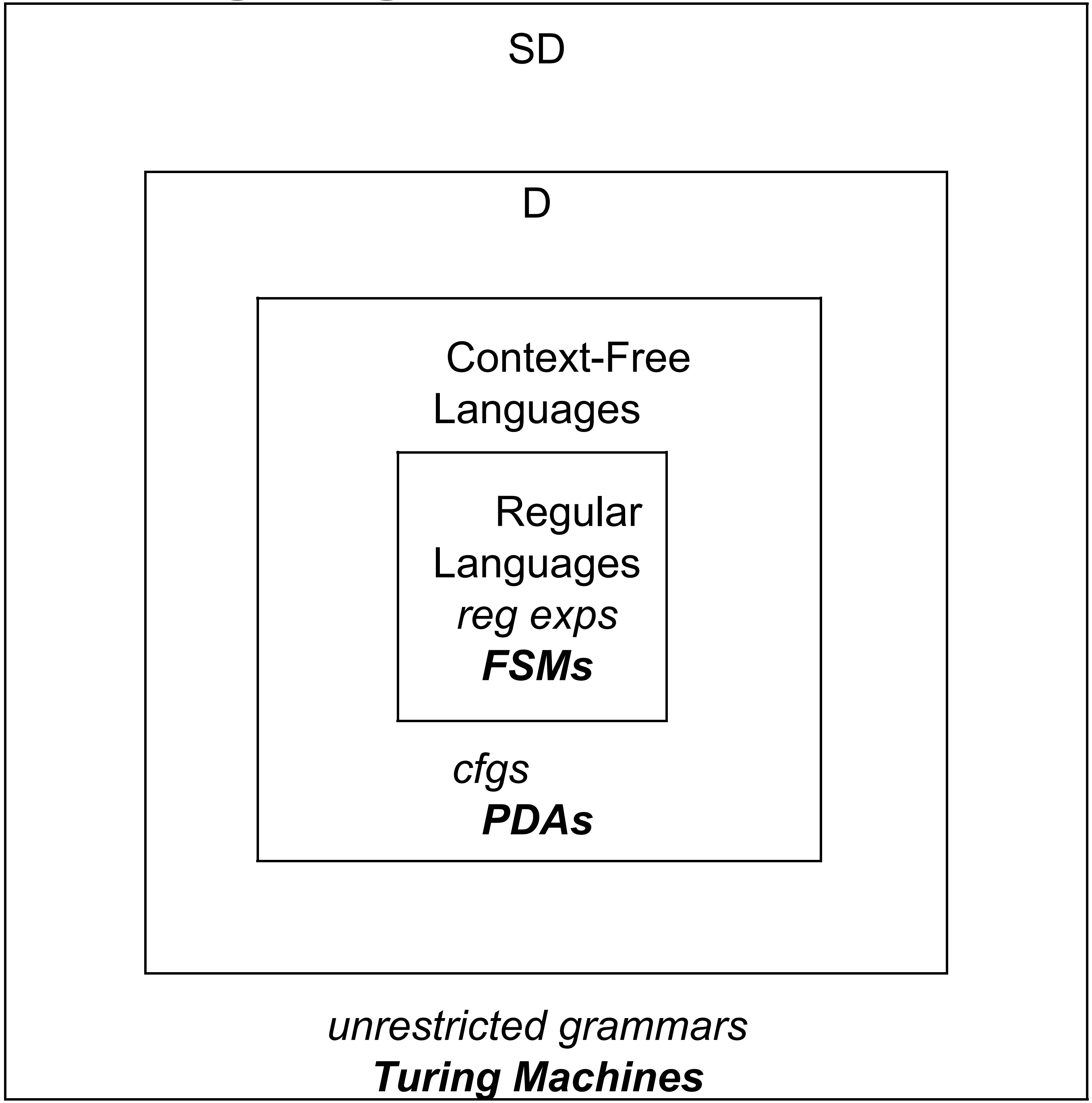
The Unsolvability of the Halting Problem

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 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:

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

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

Defining the Universe

$L1$ = {<M, w> : TM M halts on input string w}.
$L2$ = {<M> : M halts on nothing}.
$L3$ = {<Ma, Mb> : Ma and Mb halt on the same strings}.
For a string w to be in $L1$, 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:

¬*L*1 = {*x*: *x* is not a syntactically well formed <*M*, *w*> pair}

∪

{<*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:

¬*L*1 = {<*M*, *w*> : TM *M* does not halt on input string *w*}.

The Halting 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?

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 *MH* semidecides H:

$MH(\langle M, w\rangle) =$

1. Run *M* on *w*.
2. accept

MH halts iff *M* halts on *w*. Thus *MH* 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 *MH* would decide it. That is, *MH* behaves like this:

On input <*M, w*>:

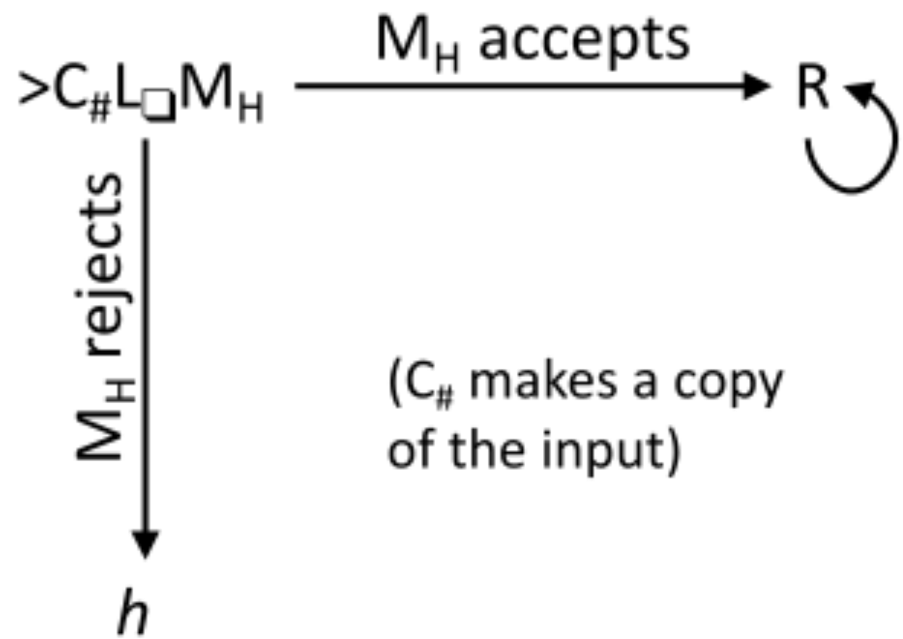
If <*M, w*> is not encoding a TM and string, then reject.

If *M* halts on input *w*

Then accept.

Else reject.

Trouble



Define *TM* *Trouble*:
On input *x*:
  Run *MH* on <*x*, *x*>
  If *MH* accepts, then loop.
  Else, halt.
(See picture at right.)

Let's run now *Trouble(<Trouble>)*. This runs *MH on* <*Trouble,Trouble*>, which decides the behavior of *Trouble* on <*Trouble*>. There are two possibilities:

1. *Trouble* halts. Then *MH* accepts, so *Trouble* loops, a contradiction.
2. *Trouble* does not halts. Then *MH* rejects, so *Trouble* halts, a contradiction.

Viewing the Halting Problem as Diagonalization

- Lexicographically enumerate Turing machines: *TMi, i = 1,2,3,…*
- Lexicographically enumerate all possible inputs: *ij, j = 1,2,3,…*
- Build the table[*i*,*j*] = 1 if *TMi* halts on *j* and 0 otherwise
- *M*H can be used to fill in any cell in the table

	i1	i2	i3	…	<Trouble>	…
TM1	1					
TM2		1				
TM3					1	
…				1		
Trouble			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 = {<*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 *ML* 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*(*ML*):

M'(*w*: string) =
1. Run *O* on <*ML*, *w*>.
2. If *O* accepts (i.e., *ML* will halt), then:
    - 2.1. Run *ML* 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 **SD** **OUT**

Semideciding TM H

D

Deciding TM AnBnCn Diagonalize

Context-Free

CF grammar AnBn Pumping

PDA Closure

Closure

Regular

Regular Expression a*b* Pumping

FSM Closure