PART 3:

Automata:

Turing Machines

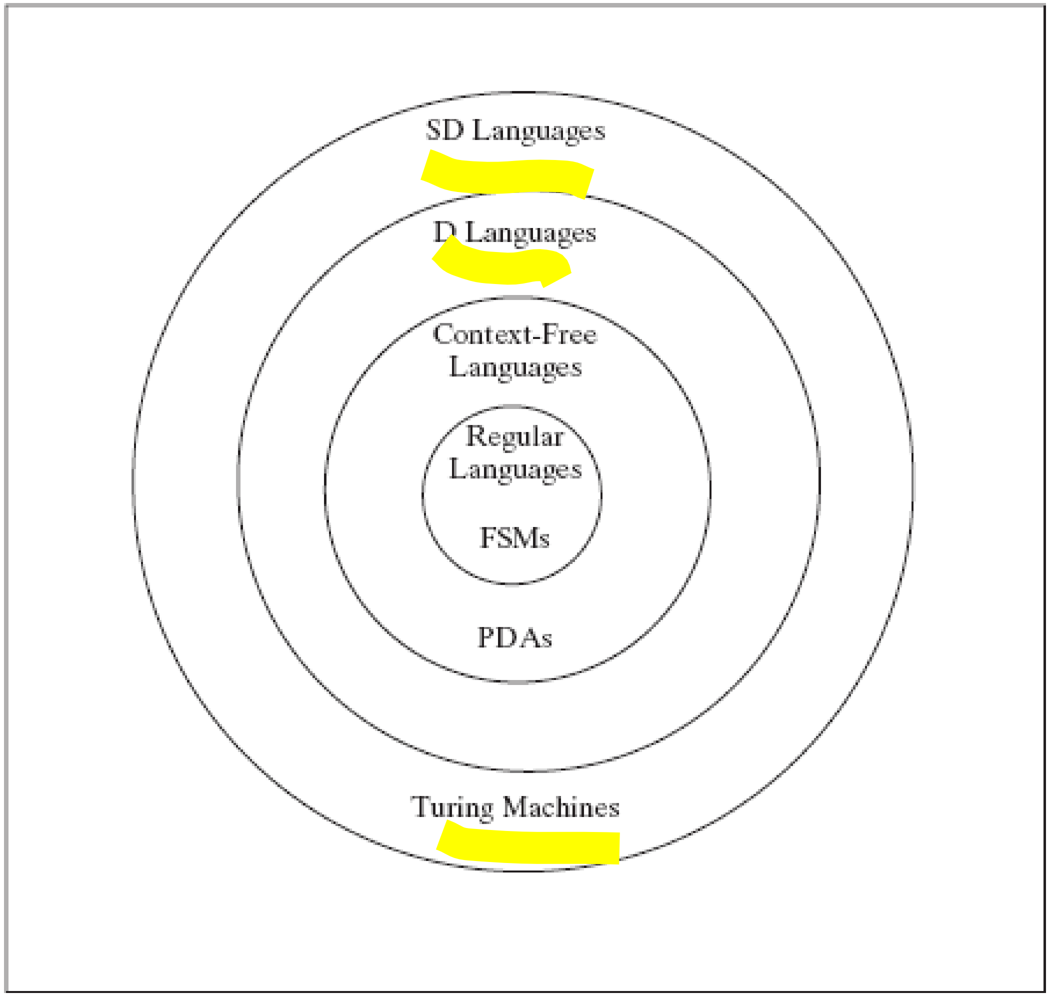
Formal Languages & Computability Theory:

Church-Turing Thesis
Unsolvability/Undecidability of the Halting Problem
Decidable & Non-Decidable Languages
Semi-Decidable & Non-Semi-Decidable Languages

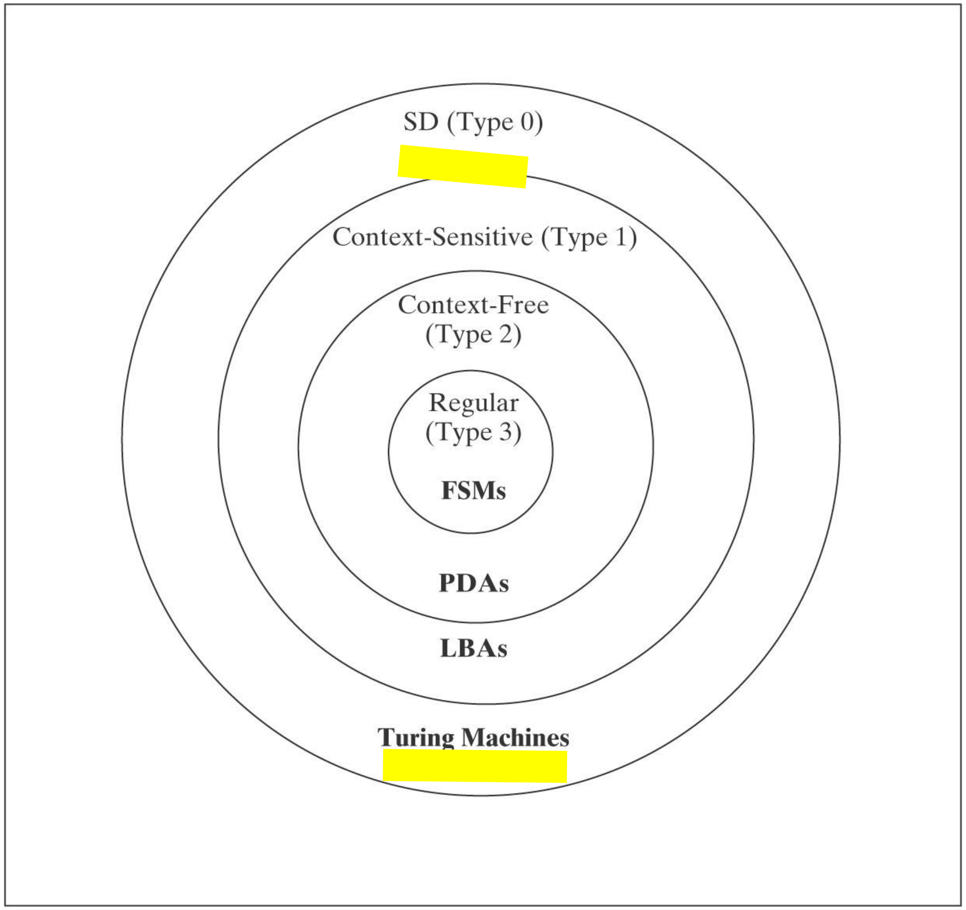
Grammar:

Unrestricted Grammars

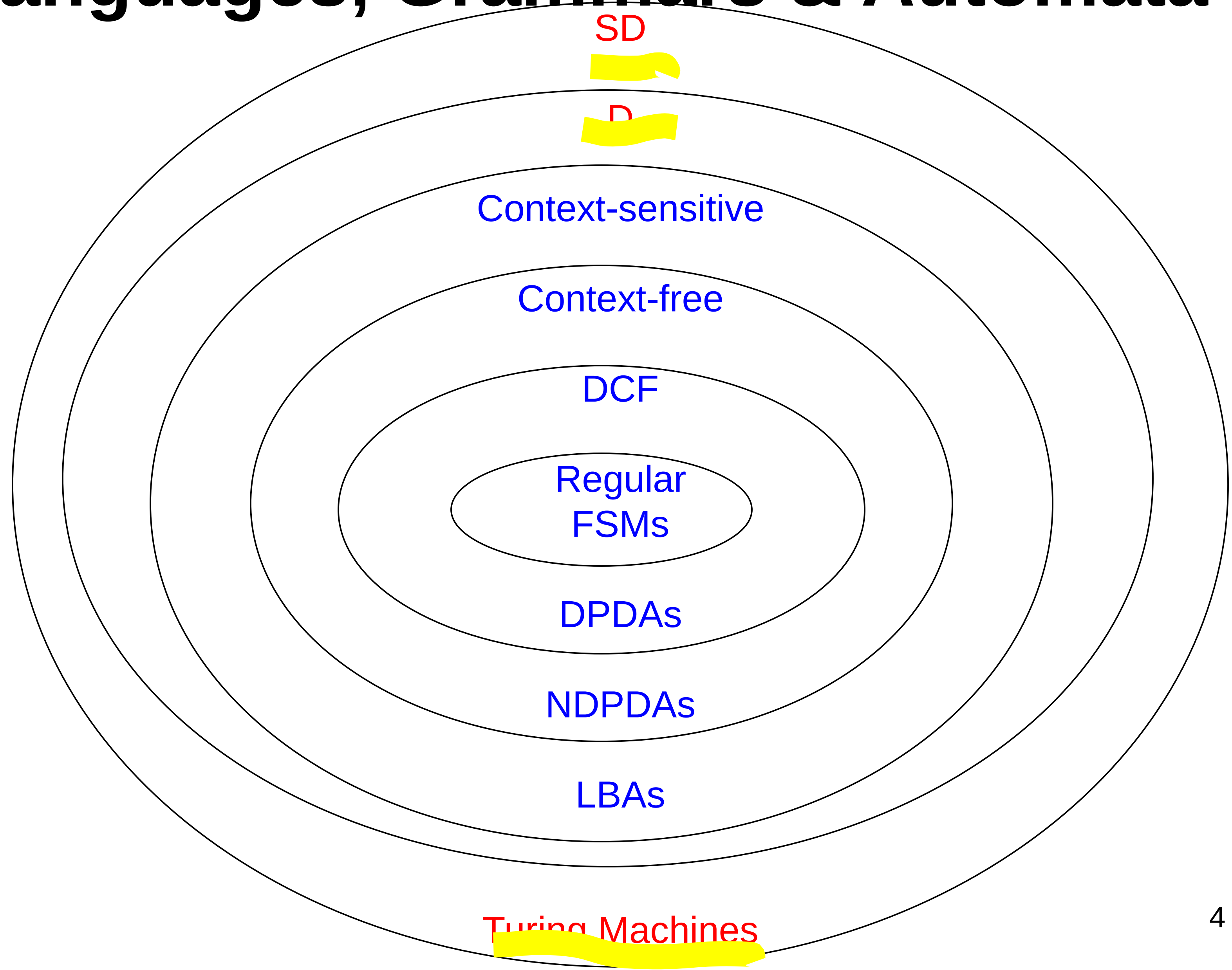
Languages, Grammars & Automata



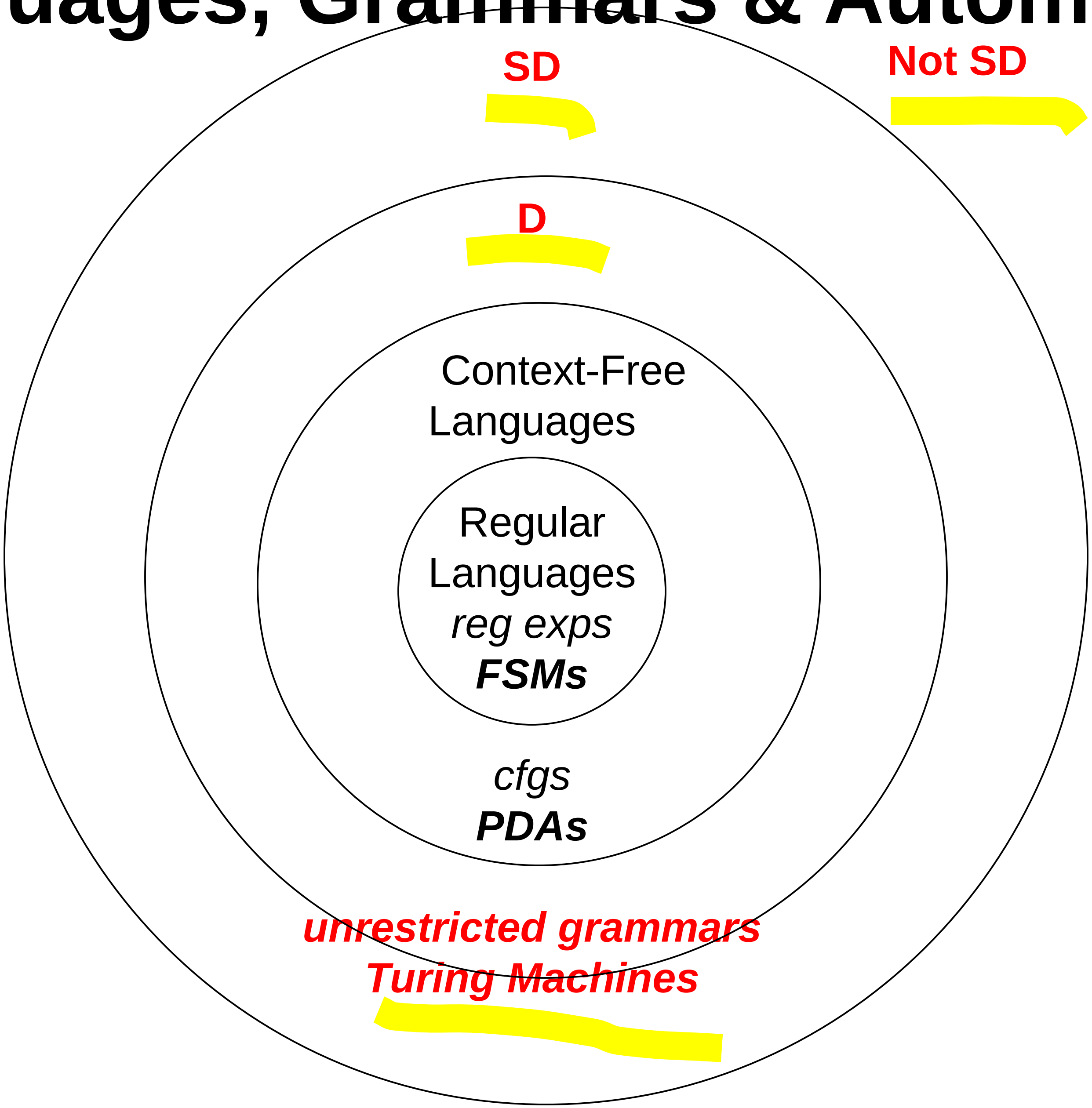
Languages, Grammars & Automata



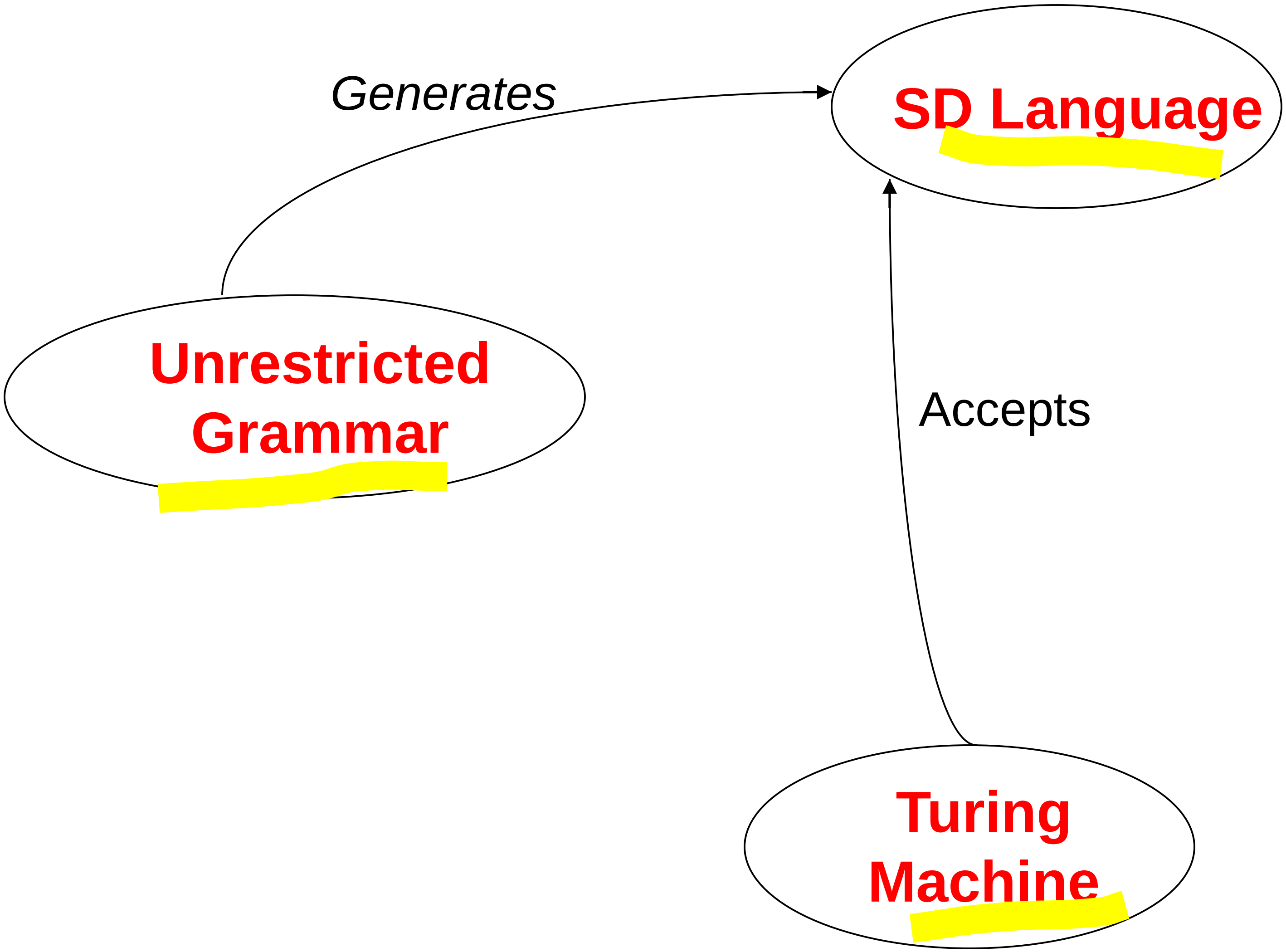
Languages, Grammars & Automata



Languages, Grammars & Automata

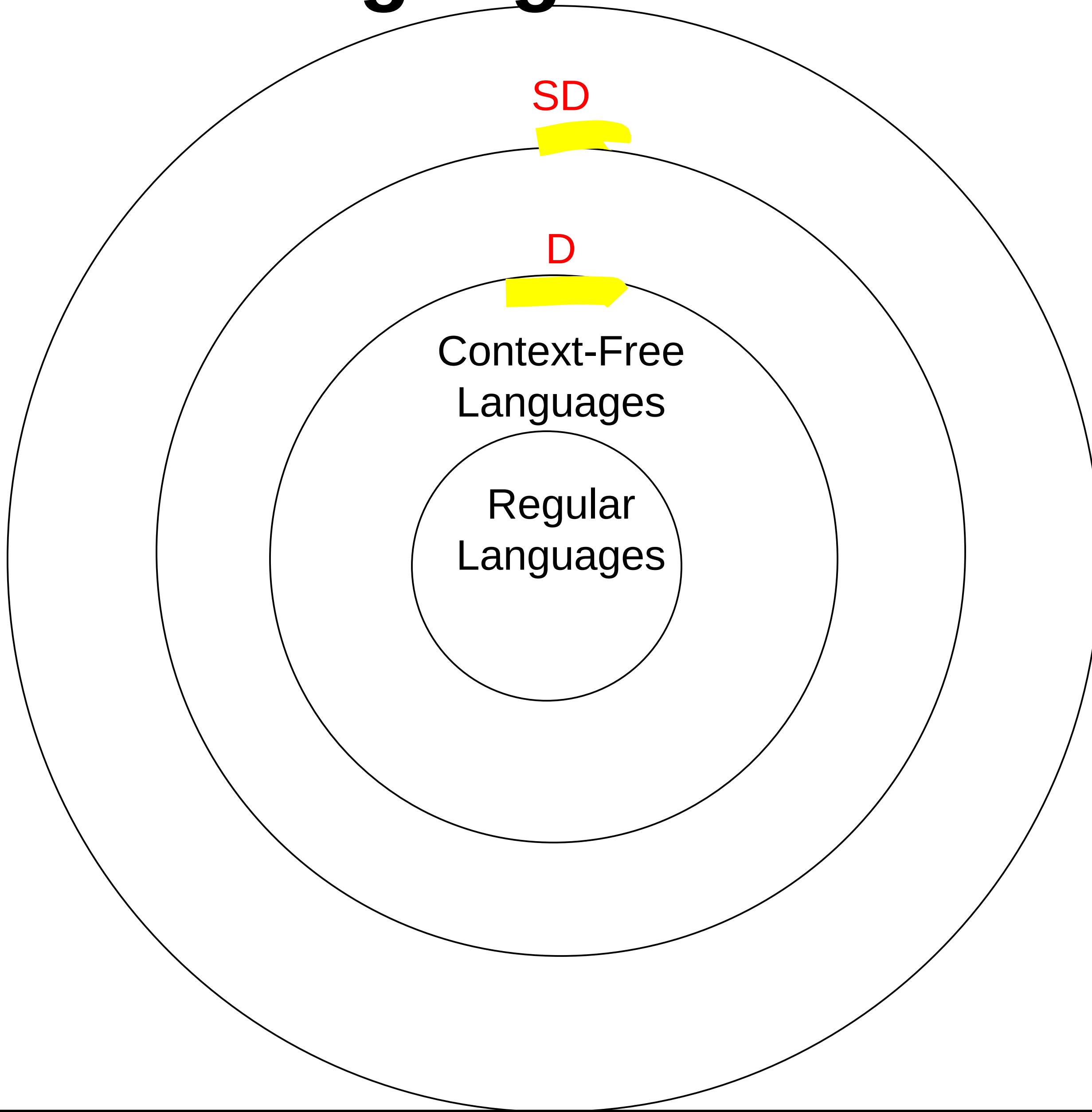


Grammars, SD Languages, and TMs



Decidable Languages
and
Semi-Decidable Languages

D and SD Languages



Decidable Languages D

- **Solvable** Languages
- **Computable** Languages
- **Recursive** Languages
- **Turing Decidable** Languages

Semi-Decidable Languages SD

- **Recursively Enumerable (R.E.)** Languages
- **Partially Decidable** Languages
- **Turing Recognizable** Languages

RL & CFL is in D

Theorem 20.1 The set of context-free languages is a *proper* subset of D.

Proof Idea:

- Every context-free language is decidable, so the context-free languages are a subset of D.
- There is at least one language, $A^nB^nC^n$, that is decidable but not context-free.

- So the context-free languages are a *proper* subset of D.

D and SD Languages

Almost every obvious language that is in SD is also in D:

- $A^nB^nC^n = \{a^n b^n c^n, n \geq 0\}$
- $\{wcw, w \in \{a, b\}^*\}$
- $\{ww, w \in \{a, b\}^*\}$
- $\{w = x*y=z: x,y,z \in \{0, 1\}^*$ and, when x, y, and z are viewed as binary numbers, $xy = z\}$

Non-D and SD Languages

But there are languages that are in SD but not in D:

- H = {<M, w> : M halts on input w}
- L = {w: w is the email address of someone who will respond to a message you just posted to your newsgroup}

D is a Subset of SD

Theorem 20.2 Every decidable language is also semidecidable.

Proof Idea:

There Exist Languages that Are Not Semi-Decidable

Theorem 20.3 There are languages that are not in SD.

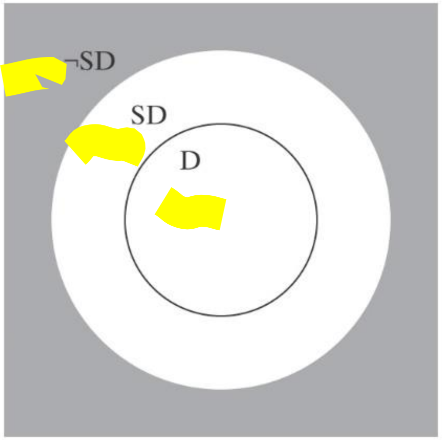
Proof Idea: Assume any nonempty alphabet Σ.

Lemma: There is a countably infinite number of SD languages over Σ.

Lemma: There is an uncountably infinite number of languages over Σ.

So there are more languages than there are languages in SD. Thus there must exist at least one language that is in $\neg$SD.

D and SD and ¬SD



1. D is a subset of SD. Every decidable language is also semidecidable.
2. There exists at least one language that is in SD/D, the donut in the picture.
3. There exist languages that are not in SD.

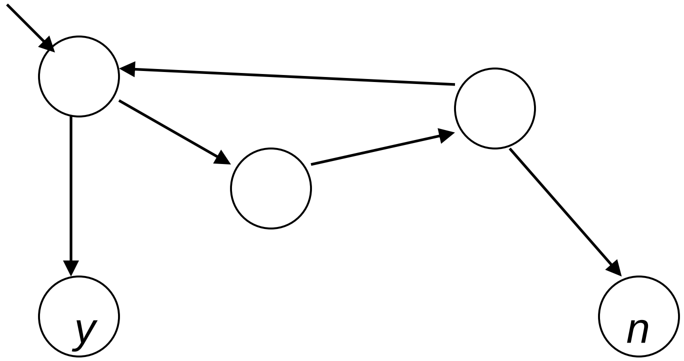
Complements of D and SD

Closure of D Under Complement

Theorem 20.4 The set D is closed under complement.

Proof Idea: Proof by construction. If *L* is in D, then there is a deterministic Turing machine *M* that decides it.

M:



From *M*, we construct *M*′ to decide $\neg L$:

Non-Closure of SD Under Complement

Theorem 20.5 The set SD is not closed under complement.

Proof Idea:

Proof by Contradiction

If so, every language in SD would also be in D.

But we know that there is at least one language (*H*) that is in SD but not in D.

Contradiction!

Property of Decidable Languages

Theorem 20.6 A language is in D iff both it and its complement are in SD.

Proof Idea:

L in D implies *L* and $\neg L$ are in SD:

- *L* is in SD because D $\subset$ SD.
- D is closed under complement
- So $\neg L$ is also in D and thus in SD.

L and $\neg L$ are in SD implies *L* is in D:

- M_1 semidecides *L*.
- M_2 semidecides $\neg L$.
- To decide *L*:
  Run M_1 and M_2 in parallel on *w*.
  Exactly one of them will eventually accept.

¬*H* is Not in SD

H is Not in D

The language **H** = {<*M*, *w*> : TM *M* halts on input string *w*} is **not decidable**.

$\neg H$ is Not in SD

Theorem 20.7 The language $\neg H$ = {<*M*, *w*> : TM *M* does not halt on input string *w*} is not in SD (or not Turing-recognizable or Turing unrecognizable).

Proof Idea:

H is in SD.
If $\neg H$ were also in SD then *H* would be in D.
But *H* is not in D.
So $\neg H$ is not in SD.

Enumerating a Language

Enumerator: Enumerating a Language

We say that Turing machine M ***enumerates*** the language L iff, for some fixed state p of M:

$$L = \{w : (s, \varepsilon) \vdash_M^* (p, w)\}$$
$$= \{w : (s, \underline{\square}) \vdash_M^* (p, w)\}$$

A language is ***Turing-enumerable*** iff there is a Turing machine that enumerates it.

SD = Turing Enumerable

Theorem 20.8 A language is **SD** iff it is **Turing-enumerable.**

Proof Idea:

Proof by Construction

Proof that Turing-enumerable implies SD:

Proof that SD implies Turing-enumerable:

Lexicographic Enumeration

M *lexicographically enumerates* *L* iff *M* enumerates the elements of *L* in lexicographic order.

A language *L* is ***lexicographically Turing-enumerable*** iff there is a Turing machine that lexicographically enumerates it.

D = Lexicographically Turing Enumerable

Theorem 20.9 A language is in **D** iff it is **lexicographically Turing-enumerable**.

Proof Idea:

Proof by Construction

Proof that D implies lexicographically TE:

Proof that lexicographically TE implies D:

Language Summary

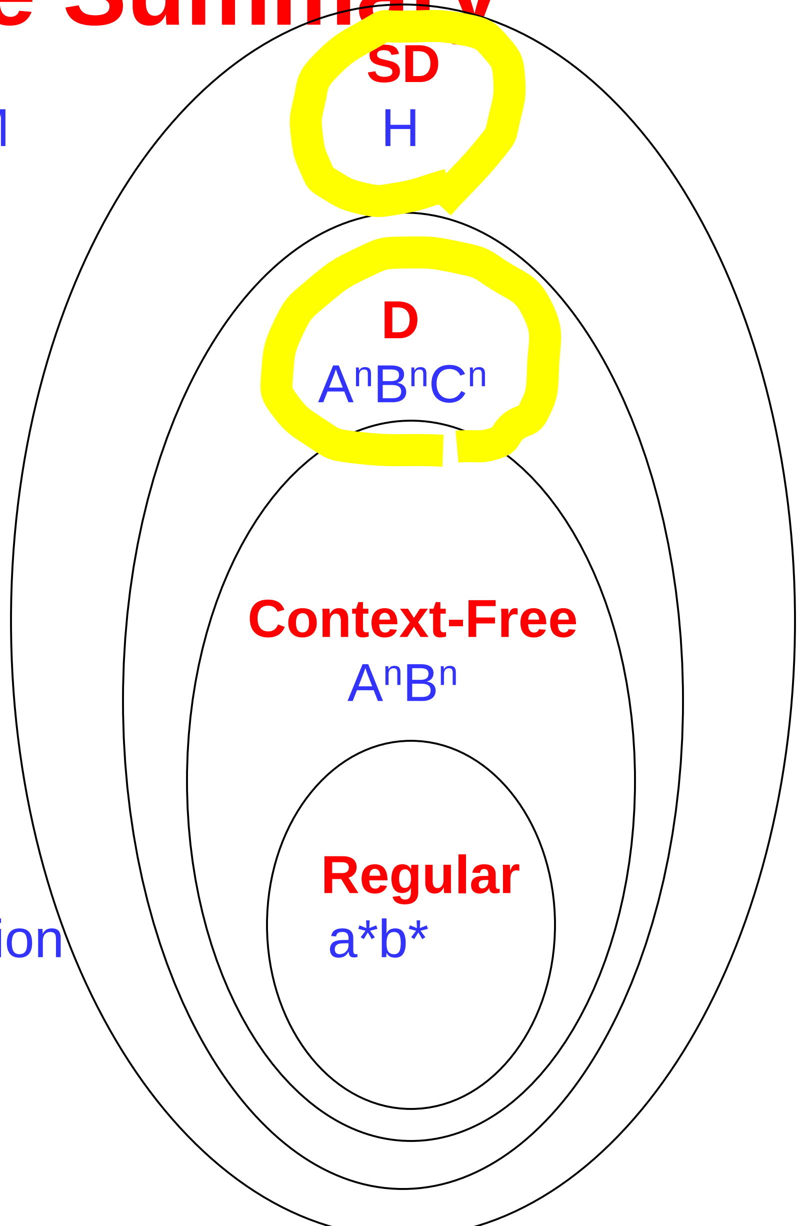
IN

Semideciding TM

Deciding TM

CF grammar
PDA

Regular Expression
FSM



OUT

Reduction

Diagonalize
Reduction

Pumping
Closure

Pumping
Closure

Reading Assignment

Chapter 20:

Sections
20.1
20.2
20.3
20.4
20.5
20.6

In-Class Exercises

Chapter 20:

1 - a
7
12
13

Non-Decidable Languages
and
Non-Semi-Decidable Languages

Two Ways to Describe a Question

- **As a language**
- **As a problem**

The Problem View and The Language View

The Problem View	The Language View
Does TM M halt on w?	H = {<M, w> : M halts on w}
Does TM M not halt on w?	¬H = {<M, w> : M does not halt on w}
Does TM M halt on the empty tape?	H_ε = {<M> : M halts on ε}
Is there any string on which TM M halts?	H_{ANY} = {<M> : there exists at least one string on which TM M halts }
Does TM M accept all strings?	A_{ALL} = {<M> : $L(M) = \Sigma^*$}
Do TMs M_a and M_b accept the same languages?	EqTMs = {<M_a, M_b> : $L(M_a) = L(M_b)$}
Is the language that TM M accepts regular?	TMreg = {<M>:$L(M)$ is regular}

Non-D Languages

There Exist Languages that Are Not Decidable

Theorem There are languages that are not in D.

Proof Idea: Assume any nonempty alphabet Σ.

Lemma: There is a countably infinite number of D languages over Σ.

Lemma: There is an uncountably infinite number of languages over Σ.

So there are more languages than there are languages in D. Thus there must exist at least one language that is in $\neg$D.

Using Mapping Reduction to Show L is not Decidable

Reduction

A ***reduction*** *R* from L_1 to L_2 is one or more Turing machines such that:

If

there exists a Turing machine *Oracle* that decides (or semidecides) L_2,

then

the Turing machines in *R* can be composed with *Oracle* to build a deciding (or a semideciding) Turing machine for L_1.

$L \leq L'$ means that L is reducible to L'.

Mapping Reductions

L_1 is ***mapping reducible*** to L_2 ($L_1 \leq_M L_2$) iff there exists some **computable function** f such that:

$$\forall x \in \Sigma^* \ (x \in L_1 \leftrightarrow f(x) \in L_2).$$

To decide whether x is in L_1, we transform it, using f, into a new object and ask whether that object is in L_2.

Using Mapping Reduction for Undecidability

(R is a reduction from L_1 to L_2) $\wedge$ (L_2 is in D) $\rightarrow$ (L_1 is in D)

If (L_1 is in D) is false,
then
at least one of the two antecedents of that implication must be false. So:

If (*R* is a reduction from L_1 to L_2) is true,
then
(L_2 is in D) must be false.

Using Mapping Reduction for Undecidability

Showing that L_2 is not in D:

L_1	(known not to be in D)	L_1 in D	But L_1 not in D
	R ↓	⇑	⇓
L_2	(a new language whose decidability we are trying to determine)	If L_2 in D	So L_2 not in D

The direction of reduction is important!

Using Mapping Reduction for Undecidability

1. Choose a language L_1:
   - that is already known not to be in D, and
   - that can be reduced to L_2.

2. Define the reduction *R.*

3. Describe the composition *C* of *R* with *Oracle*.

4. Show that *C* does correctly decide L_1 iff *Oracle* exists. We do this by showing:
   - *R* can be implemented by Turing machines,
   - *C* is correct:

     If $x \in L_1$, then $C(x)$ accepts, and

     If $x \notin L_1$, then $C(x)$ rejects.

“Does M Halt on ε?”

"Does M Halt on ε?" is SD

Theorem 21.1 H_ε = {<*M*> : TM *M* halts on ε} is in SD.

Proof Idea:

Proof by Construction

*TM **T***:

***T*(<*M*>) =**

1. Run *M* on ε.
2. Accept.

T accepts <*M*> iff *M* halts on ε, so *T* semidecides H_ε.

“Does M Halt on ε ?” is Undecidable

Theorem 21.1 H_{ε} = {<*M*> : TM *M* halts on ε} is not in D.

Proof Idea:

Proof by Contradiction By reduction from H:

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

R ↓

(?*Oracle*) H_{ε} {<*M*> : TM *M* halts on ε}

R is a mapping reduction from H to H_{ε}:

H_ε is not in D

***R*(<*M*, *w*>) =**

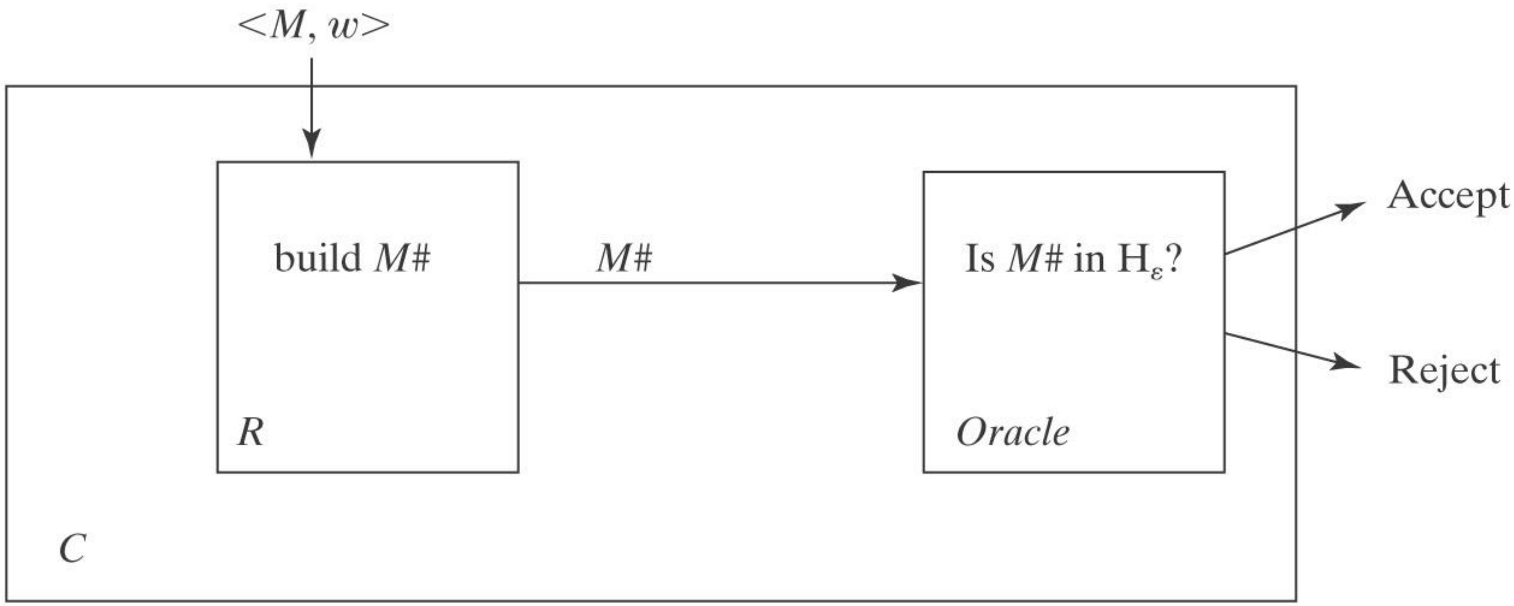
1. Construct <*M*#>, where *M*#(*x*) operates as follows:
    1.1. Erase the tape.
    1.2. Write *w* on the tape.
    1.3. Run *M* on *w*.
2. Return <*M*#>.

If *Oracle* exists, *C* = *Oracle*(*R*(<*M*, *w*>)) decides H:

C is correct: *M*# ignores its own input. It halts on everything or nothing. So:

- ✓ <*M*, *w*> $\in$ *H*: *M* halts on *w*, so *M*# halts on everything. In particular, it halts on ε. *Oracle* accepts.
- ✓ <*M*, *w*> $\notin$ *H*: *M* does not halt on *w*, so *M*# halts on nothing and thus not on ε. *Oracle* rejects.

H_ε is not in D



H_ε is not in D

- *R* can be implemented as a Turing machine.
- *C* is correct.
- So, if *Oracle* exists:

  C = *Oracle*(*R*(<*M*, *w*>)) decides H.
- But no machine to decide H can exist.
- So neither does *Oracle*.

“Does M Halt on Anything?”

"Does M Halt on Anything?" is SD, but Undecidable

Theorem 21.2 H_{ANY} = {<*M*> : there exists at least one string on which TM *M* halts} is in SD, but not in D.

Proof Idea:

H_{ANY} is in SD

Proof Idea:

Proof by Construction By exhibiting a TM *T* that semidecides it.

The Dovetailing Method

TM ***T*:**

***T*(<*M*>) =**

1. Use **dovetailing** to try *M* on all of the elements of Σ*:

ε [1]				
ε [2]	a [1]			
ε [3]	a [2]	b [1]		
ε [4]	a [3]	b [2]	aa [1]	
ε [5]	a [4]	b [3]	aa [2]	ab [1]

2. If any instance of *M* halts, halt and accept.

T will accept iff *M* halts on at least one string. So *T* semidecides H_{ANY}.

H_{ANY} is not in D

Proof Idea:

Proof by Contradiction By reduction from H:

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

R ↓

(?*Oracle*) H_{ANY} = {<*M*> : there exists at least one string on which TM *M* halts}

***R*(<*M*, *w*>)** =

1. Construct <*M#*>, where *M#*(*x*) operates as follows:
   1.1. Examine *x*.
   1.2. If *x* = *w*, run *M* on *w*, else loop.
2. Return <*M#*>.

H_{ANY} is not in D

If *Oracle* exists, then *C* = *Oracle*(*R*(<*M*, *w*>)) decides H:

C is correct: The only string on which M# can halt is w. So:

- ✓ <M, w> $\in$ H: M halts on w. So M# halts on w. There exists at least one string on which M# halts. Oracle accepts.
- ✓ <M, w> $\notin$ H: M does not halt on w, so neither does M#. So there exists no string on which M# halts. Oracle rejects.

But no machine to decide H can exist, so neither does *Oracle*.

H_{ANY} is not in D

Proof Idea:

Proof by Contradiction By reduction from H:

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

R ↓

(?*Oracle*) H_{ANY} = {<*M*> : there exists at least one string on which TM *M* halts}

***R*(<*M*, *w*>) =**

1. Construct the description <*M#*>, where *M#*(*x*) operates as follows:
    1.1. Erase the tape.
    1.2. Write *w* on the tape.
    1.3. Run *M* on *w*.
2. Return <*M#*>.

H_{ANY} is not in D

If *Oracle* exists, then *C* = *Oracle*(*R*(<*M*, *w*>)) decides H:

C is correct: *M#* ignores its own input. It halts on everything or nothing. So:

- ✓ <*M*, *w*> $\in$ H: *M* halts on *w*, so *M#* halts on everything. So it halts on at least one string. *Oracle* accepts.
- ✓ <*M*, *w*> $\notin$ H: *M* does not halt on *w*, so *M#* halts on nothing. So it does not halt on at least one string. *Oracle* rejects.

But no machine to decide H can exist, so neither does *Oracle*.

“Does M Halt on Everything?”

"Does M Halt on Everything?" is Undecidable

Theorem 21.3 H_{ALL} = {<*M*> : TM *M* halts on all inputs} is not in D.

Proof Idea:

Proof by Contradiction By reduction from H:

H_{ALL} is Not in D

H_ε = {<*M*> : TM *M* halts on ε}

R ↓

(?*Oracle*) H_{ALL} = {<*M*> : TM *M* halts on all inputs }

***R*(<*M*>) =**

1. Construct the description <*M#*>, where *M#*(*x*) operates as follows:
    1.1. Erase the tape.
    1.2. Run *M*.
2. Return <*M#*>.

If *Oracle* exists, then *C* = *Oracle*(*R*(<*M*>)) decides H_ε:

- *R* can be implemented as a Turing machine.
- *C* is correct: *M#* halts on everything or nothing, depending on whether *M* halts on ε. So:
  - ✓ <*M*> $\in H_\varepsilon$: *M* halts on ε, so *M#* halts on all inputs. *Oracle* accepts.
  - ✓ <*M*> $\notin H_\varepsilon$: *M* does not halt on ε, so *M#* halts on nothing. *Oracle* rejects.

But no machine to decide H_ε can exist, so neither does *Oracle*.

“Does M accept w?”

“Does M accept w?” is Undecidable

Theorem 21.4 A = {<*M*, *w*> : *M* accepts *w and* $w \in L(M)$} is not in D.

Proof Idea:

Proof by Contradiction By reduction from H:

A = {<*M*, *w*> : *w* ∈ *L*(*M*)} is Not in D

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

R ↓

(?*Oracle*) A = {<*M*, *w* > : *w* ∈ *L*(*M*) }

***R*(<*M*, *w*>) =**

1. Construct the description <*M#*>, where *M#*(*x*) operates as follows:
   1.1. Erase the tape.
   1.2. Write *w* on the tape.
   1.3. Run *M* on *w*.
   1.4. **Accept**
2. Return <*M#, w*>.

If *Oracle* exists, then *C* = *Oracle*(*R*(<*M*, *w*>)) decides H:

- *R* can be implemented as a Turing machine.
- *C* is correct: *M#* accepts everything or nothing. So:
  - ✓ <*M, w*> ∈ H: *M* halts on *w*, so *M#* accepts everything. In particular, it accepts *w*. *Oracle* accepts.
  - ✓ <*M, w* > ∉ H: *M* does not halt on *w*. *M#* gets stuck in step 1.3 and so accepts nothing. *Oracle* rejects.

But no machine to decide H can exist, so neither does *Oracle*.

A_ε, A_{ANY}, and A_{ALL} are Undecidable

Theorem 21.5 A_ε = {<*M*> : TM *M* accepts ε} is not in D.

Proof Idea: Analogous to that for H_ε.

Theorem 21.6 A_{ANY} = {<*M*> : TM *M* accepts at least one string} is not in D.

Proof Idea: Analogous to that for H_{ANY}.

Theorem A_{ALL} = {<*M*> : = *L*(*M*) = Σ*} is not in D.

Proof Idea: Analogous to that for H_{ALL}.

“Are Two TMs Equivalent ?”

“Are Two TMs Equivalent?” is Undecidable

Theorem 21.8 EqTMs={<M_a, M_b>: $L(M_a)=L(M_b)$} is not in D.

Proof Idea:

Proof by Contradiction By reduction from A_{ALL} :

EqTMs={<M_a, M_b>: $L(M_a)=L(M_b)$} is Not in D

A_{ALL} = {<*M*> : $L(M) = \Sigma^*$}

R ↓

(*Oracle*) EqTMs = {<M_a, M_b>: $L(M_a)=L(M_b)$}

***R*(<*M*>) =**

1. Construct the description of *M#*(*x*):
   1.1. Accept.
2. Return <*M*, *M#*>.

If *Oracle* exists, then *C* = *Oracle*(*R*(<*M*>)) decides A_{ALL}:

- *C* is correct: *M#* accepts everything. So if *L*(*M*) = *L*(*M#*), *M* must also accept everything. So:
  - ✓ <*M*> $\in$ A_{ALL}: *L*(*M*) = *L*(*M#*). *Oracle* accepts.
  - ✓ <*M*> $\notin$ A_{ALL}: *L*(*M*) $\neq$ *L*(*M#*). *Oracle* rejects.

But no machine to decide A_{ALL} can exist, so neither does *Oracle*.

Are All Questions about TMs Undecidable?

Example 21.8

L = {<M> : TM M contains an even number of states}

Example 21.9

L = {<M, w> : M halts on w within 3 steps}.

Rice's Theorem

Property of the SD language

A *nontrivial property* of the SD language is one that is not simply:

- *True* for all languages,

  or
- *False* for all languages.

Rice's Theorem

Theorem 21.10 No nontrivial property of the SD languages is decidable.

or

Every nontrivial property of the SD languages is undecidable.

or

Any language that can be described as:
{<M>: $P(L(M))$ = *True*} for any nontrivial property P, is not in D.

Applying Rice's Theorem

To use Rice's Theorem to show that a language *L* is not in D we must:

- Specify property *P*.
- Show that the domain of *P* is the SD languages.
- Show that *P* is nontrivial: *P* is true of at least one language & *P* is false of at least one language.

Applying Rice's Theorem?

- L = {<*M*> : $L(M)$ contains only even length strings}.
- L = {<*M*> : $L(M)$ contains an odd number of strings}.
- L = {<*M*> : $L(M)$ contains all strings that start with a}.
- L = {<*M*> : $L(M)$ is infinite}.
- L = {<*M*> : $L(M)$ is regular}.
- L = {<*M*> : *M* contains an even number of states}.
- L = {<*M*> : *M* has an odd number of symbols in its tape alphabet}.
- L = {<*M*> : *M* accepts ε within 100 steps}.
- L = {<*M*>: *M* accepts ε}.
- L = {<M_a, M_b> : $L(M_a) = L(M_b)$}.

"Is $L(M)$ Regular?"

“Is *L*(*M*) Regular?” is Undecidable

Theorem 21.11 TMreg{<*M*> : *L*(*M*) is regular} is not in D?

Proof Idea:

By Rice’s Theorem:

- *P* = *True* if *L* is regular and *False* otherwise.
- The domain of *P* is the set of SD languages since it is the set of languages accepted by some TM.
- *P* is nontrivial:
  - *P*(a*) = *True*.
  - *P*(A^nB^n) = *False*.

Non-SD Languages

There Exist Languages that Are Not Semi-Decidable

Theorem 20.3 There are languages that are not in SD.

Proof Idea: Assume any nonempty alphabet Σ.

Lemma: There is a countably infinite number of SD languages over Σ.

Lemma: There is an uncountably infinite number of languages over Σ.

So there are more languages than there are languages in SD. Thus there must exist at least one language that is in $\neg$SD.

Non-SD Languages

Intuition: Non-SD languages usually involve either infinite search or knowing a TM will infinite loop.

Examples:

- $\neg H = \{<M, w> : \text{TM } M \text{ does } \textit{not} \text{ halt on } w\}$.
- $L = \{<M> : L(M) = \Sigma^*\}$.
- $L = \{<M> : \text{TM } M \text{ halts on nothing}\}$.

Proving that Languages are not SD

- ✓ Contradiction/ *L* is the complement of an SD/D Language.
- ✓ Reduction from a known non-SD language

"Does There Exist No String on which M Halts?"

“Does There Exist No String on which M Halts?” is Not SD

Theorem 21.15 $H_{\neg ANY}$ = {<*M*> : there does ***not*** exist any string on which TM *M* halts} is not in SD (or not Turing-recognizable or Turing unrecognizable).

Proof Idea:

Proof by Contradiction

$\neg H_{\neg ANY}$ is H_{ANY} where

H_{ANY} = {<*M*> : there exists at least one string on which TM *M* halts}.

We already know:

- $\neg H_{\neg ANY}$ is in SD.
- $\neg H_{\neg ANY}$ is not in D.

So $H_{\neg ANY}$ is not in SD because, if it were, then H_{ANY} would be in D but it isn’t.

Using Reduction for Unsemidecidability

If there is a reduction *R* from L_1 to L_2 and L_1 is not SD, then L_2 is not SD.

So, we must:

- Choose a language L_1 that is known not to be in SD.
- Hypothesize the existence of a *semideciding* TM *Oracle*.

“Does There Exist No String on which M Halts?” is Not SD

Theorem 21.15 ***Proof Idea:***

Proof by Contradiction By reduction from ¬ H:

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

R ↓

(?*Oracle*) $H_{\neg ANY}$ = {<*M*> : there does not exist a string on which TM *M* halts}

***R*(<*M*, *w*>) =**

1. Construct the description <*M#*> of *M#*(*x*):
   1.1. Erase the tape.
   1.2. Write *w* on the tape.
   1.3. Run *M* on *w*.
2. Return <*M#*>.

“Does There Exist No String on which M Halts?” is Not SD

If *Oracle* exists, then *C* = *Oracle*(*R*(<*M*, *w*>)) semidecides ¬H:

- *C* is correct: *M#* ignores its input. It halts on everything or nothing, depending on whether *M* halts on *w*. So:
  - ✓ <*M*, *w*> ∈ ¬H: *M* does not halt on *w*, so *M#* halts on nothing. *Oracle* accepts.
  - ✓ <*M*, *w*> ∉ ¬H: *M* halts on *w*, so *M#* halts on everything. *Oracle* does not accept.

But no machine to semidecide ¬H can exist, so neither does *Oracle*.

Summary of D, SD/D or ¬ SD?

D, SD/D or ¬ SD?

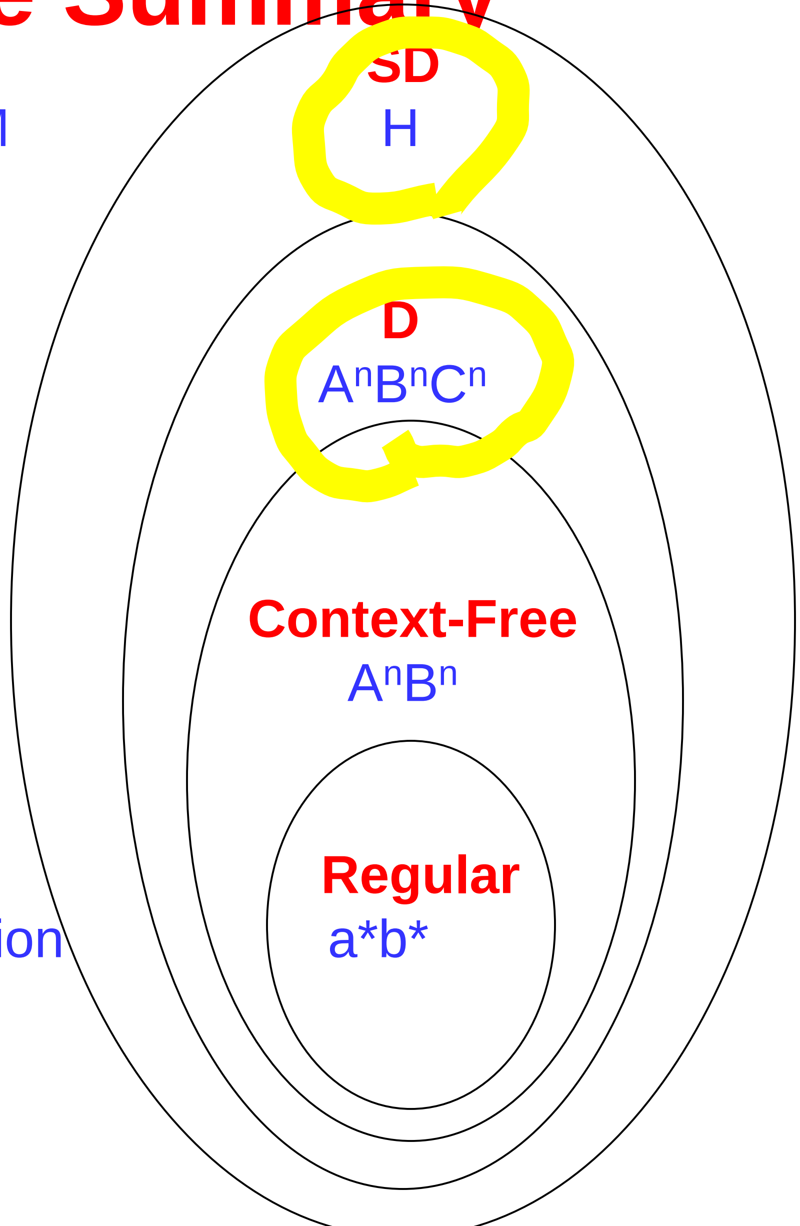
- **Decidable Languages D**
  - Solvable Languages
  - Computable Languages
  - Recursive Languages
  - Turing Decidable Languages
- **Semi-Decidable Languages SD**
  - Recursively Enumerable (R.E.) Languages
  - Partially Decidable Languages
  - Turing Recognizable Languages
- ¬ **D** Turing Undecidable Languages
- ¬ **SD** Turing Unrecognizable Languages

The Problem View	***The Language View***	***Status***
Does TM *M* have an even number of states?	{<*M*> : *M* has an even number of states}	D
Does TM *M* halt on *w*?	H = {<*M*, *w*> : *M* halts on *w*}	SD/D
Does TM *M* halt on the empty tape?	H_ε = {<*M*> : *M* halts on ε}	SD/D
Is there any string on which TM *M* halts?	H_{ANY} = {<*M*> : there exists at least one string on which TM *M* halts }	SD/D
Does TM *M* halt on all strings?	H_{ALL} = {<*M*> : *M* halts on Σ*}	¬SD
Does TM *M* accept *w*?	A = {<*M*, *w*> : *M* accepts *w*}	SD/D
Does TM *M* accept ε?	A_ε = {<*M*> : *M* accepts ε}	SD/D
Is there any string that TM *M* accepts?	A_{ANY} {<*M*> : there exists at least one string that TM *M* accepts }	SD/D

Does TM M accept all strings?	$A_{ALL} = \{<M> : L(M) = \Sigma^*\}$	¬SD
Do TMs M_a and M_b accept the same languages?	$EqTMs = \{<M_a, M_b> : L(M_a) = L(M_b)\}$	¬SD

Does TM M not halt on any string?	$H_{\neg ANY} = \{<M>$: there does not exist any string on which M halts$\}$	¬SD
Does TM M not halt on its own description?	$\{<M>$: TM M does not halt on input $<M>\}$	¬SD
Is TM M minimal?	$TM_{MIN} = \{<M>: M$ is minimal$\}$	¬SD
Is the language that TM M accepts regular?	$TMreg = \{<M> : L(M)$ is regular$\}$	¬SD
Does TM M accept the language A^nB^n?	$A_{anbn} = \{<M> : L(M) = A^nB^n\}$	¬SD

Language Summary



IN	OUT
Semideciding TM	Reduction
Deciding TM	Diagonalize Reduction
CF grammar PDA	Pumping Closure
Regular Expression FSM	Pumping Closure

Reading Assignment

Chapter 21:

Sections
21.1
21.2
21.3
21.4
21.6
21.7

In-Class Exercises

Chapter 21:

1 – c & i
4
5 - a
9 - b
11 – a & d
14