Chapter 5
Decidability

We have introduced Turing Machine as a "standard" model for the general computation and defined the informal notion of algorithms in terms of TM.

We are now ready to investigate the power of such a model in terms of its *computability,* namely, what can be done ☺, and what cannot be done, with this model. ☹

For example, we will show that there exists an algorithm that will decide if a CFL will generate a specific string, which is the centerpiece of a compiler, i.e., the *acceptance problem* dealing with the context-free language.

Since every regular language is context free, we immediately know that the acceptance problem dealing with regular languages is solvable.

The other side

We will also show that there doesn't exist any algorithm, or equivalently, a Turing machine, by the *Church-Turing Thesis,* that will tell us if an arbitrary TM will accept an arbitrary string.

That is, the acceptance problem dealing with a universal language, or TM, called the *Halting problem,* is unsolvable. ☹

This opens up the can of worm... in the sense that a whole collection of unsolvable problems will arise.

How could we show that? We will explore such a technique of *reducibility* in the next chapter.

Once a problem is shown algorithmically unsolvable, we have to consider its simplification, often an *approximation algorithm*, which leads us to a better understanding of its nature, as well as a good, but not best, solution at a reasonable cost.

Decidable problems w.r.t. RL

The *acceptance problem for* DFAs, i.e., whether a DFA accepts a given string, can be expressed as the following language:

$A_{\mathsf{DFA}} = \{\langle B, w\rangle : B \text{ is a DFA that accepts } w.\}$

The notions of A_{NFA} and A_{REX} are similarly defined.

Theorem: A_{DFA} is decidable.
Proof: For any input $\langle B, w\rangle$, where B is a DFA and w is a string, construct the following Turing machine M_{DFA} :
1. Simulate B on w.
2. If the simulation ends in an accept state, accept it; otherwise, reject it. □

Because we can mechanically convert any NFA, or a regular expression, REX, to a DFA, we have the following result:

Theorem: Both A_{NFA}, and A_{REX} are decidable languages.

Will this DFA accept nothing?

Let E_{DFA} be $\{\langle A \rangle : A \text{ is a DFA and } L(A) = \emptyset\}$.

Theorem: E_{DFA} is decidable.

Proof: A DFA accepts a string if and only if it is possible to go from the start state to an accept state. Thus, we can construct the following TM:

For any input $\langle A \rangle$, where A is a DFA,

1. Mark the start state of A.

2. Repeat Step 3 until no new state gets marked

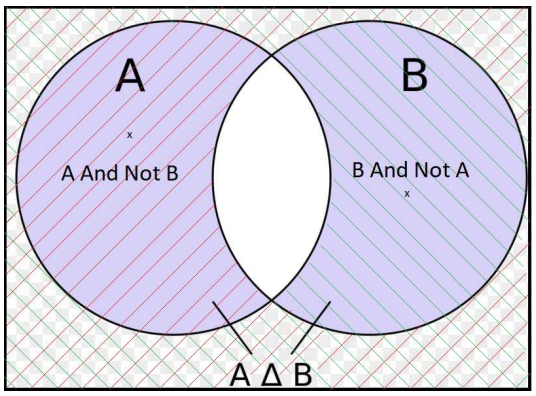
3. Mark any state that has a transition coming into *it* from a marked state.

4. If no accept state gets marked, accept ($L(A) = \emptyset$); otherwise, reject ($L(A) \neq \emptyset$).

Symmetric difference

Let both A and B be two sets, the *set difference* of A and B, denoted by $A\Delta B$, is defined as follows:

$$A\Delta B = \{x : x \in (A \cup B) \setminus (A \cap B)\}$$



Notice that $A\Delta B = (A \cup B) \setminus (A \cap B)$. It is clear that $A\Delta B = \emptyset$ iff $A \cup B = A \cap B$ iff $A = B$.

We are now ready to show that the problem that if two regular languages are the same is also decidable. ☺

Question: Any idea where we are going?

Are these two really the same?

Let EQ_{DFA} be $\{\langle A, B\rangle : A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$.

Theorem: EQ_{DFA} is decidable.

Proof: We will construct another DFA C, such that C only accepts those strings that are accepted by either A or B, but not by both.

Let $L(C) = A\Delta B = (A \cap \overline{B}) \cup (B \cap \overline{A})$, as we just saw, $L(C) = \emptyset \equiv L(A) = L(B)$.

Because both $L(A)$ and $L(B)$ are RL's, and RLs are closed under union, intersection, as well as complementation (Cf. Theorem 1.45, Results 1 and 2, FA notes), so is $L(C)$.

To decide if $L(A) = L(B)$, we simply construct C, a DFA, run E_{DFA} (Cf. Page 4) on C, and make a decision accordingly. ☺

Homework: Exercises 4.2 and 4.3.

Decidable problems w.r.t. CFL

Let $A_{\mathsf{CFG}} = \{\langle G, w\rangle : G$ is a CFG that accepts $w.\}$.

Theorem: A_{CFG} is a decidable language.

Proof: The idea to try all derivations does not work, in general, since if G doesn't generate w, the algorithm might never halt. ☹

On the other hand, we can always convert G to a grammar in CNF (Chomsky Normal Form), which can be done effectively (Cf. Pages 18 through 23, NPDA chapter), although not yet by *JFlap*.

If the latter CNF generates w, it must generate w in exactly $2|w| - 1$ steps (Cf. Exercise 2.26).

Thus, we can come up with a TM, by the *Church-Turing Thesis.*

Question: Big deal, so what?

An "inefficient" compiler

For any input $\langle G, w \rangle$, where G is a CFG and w is a string:

1. Convert G to G', in Chomsky normal form.

2 Try all the derivations of w in G', in $2|w| - 1$ steps. ☹

3. If any of them accepts w, accept *it*; otherwise, reject.

Although Step 2 is extremely time consuming, it generates and checks at most $b^{2|w|-1}$ strings, thus can be done within *a finite amount of time.*

Thus, a compiler is feasible. ☺ The rest is just an engineering story that leads a much more efficient compiler, which must be told in a course on compiler. ☺

ACM Turing Award 2020 has gone to this compiler related business (Course page).

Is this language useful?

Let E_{CFG} be $\{\langle G \rangle : G \text{ is a CFG and } L(G) = \emptyset\}$.

Theorem: E_{CFG} is decidable.

Proof: The following is an algorithm, similar to what we did to solve the same problem for RL:

For any input $\langle G \rangle$, a CFG,

1. Mark all terminals in G.

2. Repeat Step 3 until no new variable gets marked.

3. Mark any variable A, where G contains a rule $A \rightarrow U_1 \cdots U_k$, and all U_i's have been marked.

4. If the start symbol is not marked, accept; otherwise, reject. $\square$

Are they the same?

Let EQ_{CFG} be $\{\langle G, H\rangle : G$ and H are CFGs and $L(G) = L(H)\}$.

We can't use the same idea as we used to show that EQ_{DFA} is decidable (Pages 5 and 6), since CFL is not closed under either intersection or complementation (Cf. Page 83, CFL notes).

In fact, EQ_{CFG} *is not decidable.* ☹

Homework: Exercises 4.4, and 4.13.

Is this result necessary?

Theorem: Every CFL is decidable.

This result means that, for any CFL, L, and any string w, there is a TM that decides if $w \in L$. Thus,it is pretty similar to the one that we talked about earlier regarding context-free grammar.

Proof: Let G be a CFG for L, and let S be the TM we used to decide the acceptance problem for CFL (Cf. Page 10), we construct a TM, M_G : For input $\langle G, w \rangle$,

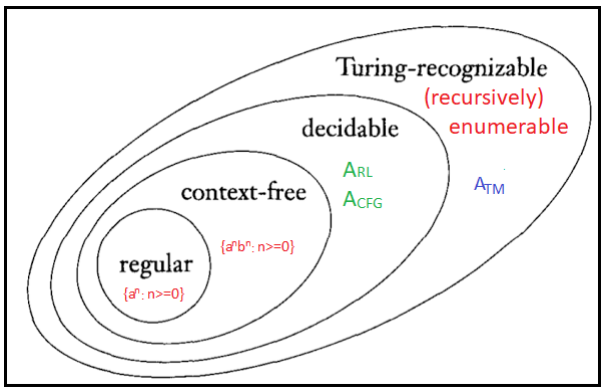
1. Run TM S on $\langle G, w \rangle$.

2. If S accepts $\langle G, w \rangle$, accept $\langle L, w \rangle$; otherwise, reject.

Since every regular language is context-free, every regular language is decidable, as well.

Some positive results

So far, we have coined four classes of languages: regular, context-free, decidable, and enumerable. The following chart shows their relationship. (See Page 89 of NPDA notes for much more details. ☺)



Question: Why the "$\subset$" relationship looks like this?

Question: Why not "$\subseteq$"?

Both A_{FA} and A_{NPDA} are decidable based on resuls as shown on Pages 3 and 9, respectively.

Question: How about A_{LBA} and A_{TM}?

The Halting problem

Since the *CS 3221* days, we have been talking about the existence of a concrete, and easily understood, problem that is algorithmically unsolvable, i.e., not decidable. ☹

See the very first page of the *Overview* notes.

We now will finally talk about *it* ☺ in detail. ☹

By the *Halting problem,* we mean the following: *Given a Turing Machine,* M*, and an input string,* w*, whether* M *accepts* w?

Let $A_{\mathsf{TM}} = \{\langle M, w\rangle : M \text{ accepts } w\}$.

Theorem: A_{TM} is undecidable.

In other words, there does not exist a TM which will decide A_{TM}, i.e., it is able to tell whether M accepts w. ☹

It is enumerable....

...since there is TM that can *accept* A_{TM}.

For example, we can construct the following TM, U, the *universal Turing Machine,* for input $\langle M, w \rangle$, where M is a TM and w is an input string:

1. Simulate M on input w,

2. If M ever enters its accept state, accept $\langle M, w \rangle$; if M ever enters its reject state, reject $\langle M, w \rangle$.

Notice that this universal machine that Turing suggested in 1936 shows the feasibility of a stored program computer, U, which runs *any* program M.

Such a computer was eventually implemented in the 1940's, the rest is just history. ☺

How could we...

... prove that A_{TM} is not decidable?

We will look at the pigeons again... ☺

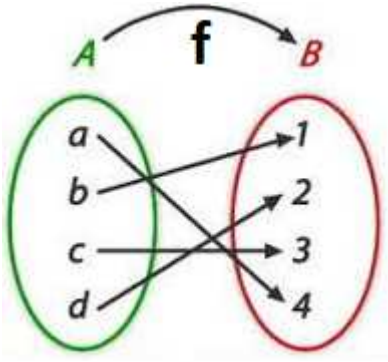
If $k + 1$ pigeons are put into k boxes, at least one box holds at least two pigeons.

Are they of the same size?

Given two finite sets, it is easy to see which is larger.

Question: How could we compare the sizes of two infinite sets?

Georg Cantor (1845-1918) observed that, if two finite sets are of the same size, then there exists a 1-1 correspondence between the elements of these two sets.



Question: Big deal?

Cantor proposed to use the same criterion for infinite sets. Check out the course page for more details about Cantor and his stuff.

How to do that?

We need to identify some basic properties of a function.

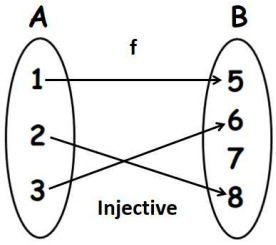
For a function $f : A \mapsto B$, $a = b \Rightarrow f(a) = f(b)$.

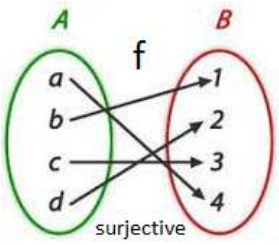
By saying that f is *one-to-one (injective)*, we mean that f never maps two different elements to the same element, i.e.,

$$\forall a \in A, b \in B \ a \neq b \Rightarrow f(a) \neq f(b).$$

Moreover, f is *onto (surjective)*, if f hits every element in B, i.e.,

$$\forall b \in B \ \exists a \in A \ f(a) = b.$$





$|A| = |B|$

Let A be a set, $|A|$ is called its *cardinality,* i.e., its size.

Question: Let A, B be two infinite set. When $|A| = |B|$, i.e., A and B are of the *same size*?

Cantor's answer: If there is a one to one, and onto function $f : A \mapsto B$.

By the pigeonhole principle, if $|A| > |B|$, for some $b \in B$, there exist at least two elements from A that will be mapped to b. This violates the assumption that f is injective, i.e., one-to-one. Hence, $|A| \leq |B|$.

Just assume $|A| < |B|$. Then, since f is a function, f maps any element in A to at most one element in B. Hence, there must exist at least one $b \in B$, such that no element from A is mapped to b, which violates the onto condition.

Hence, it must be the case that $|A| = |B|$. ☺

An example

Let $\mathcal{N}$ be the set of natural numbers, $\{1, 2, \ldots\}$ and let $\mathcal{E}$ be the set of all the even natural numbers, $\{2, 4, \ldots\}$.

Question: Which set contains more numbers?

An incorrect answer: Intuitively, $\mathcal{N}$ does, since besides even numbers, it also contains odd numbers.

A correct one: If we define a function $f : \mathcal{N} \mapsto \mathcal{E}$, $\forall n \in \mathcal{N}, f(n) = 2n$.

The function f is injective: If $\forall n_1 \neq n_2$, then clearly $f(n_1) = 2n_1 \neq 2n_2 = f(n_2)$.

The function is also surjective: Let $m = 2n \in \mathcal{E}$, just pick $n \in \mathcal{N}$, we have $f(n) = 2n = m$.

Hence, by definition, $|\mathcal{N}| = |\mathcal{E}|$. ☹

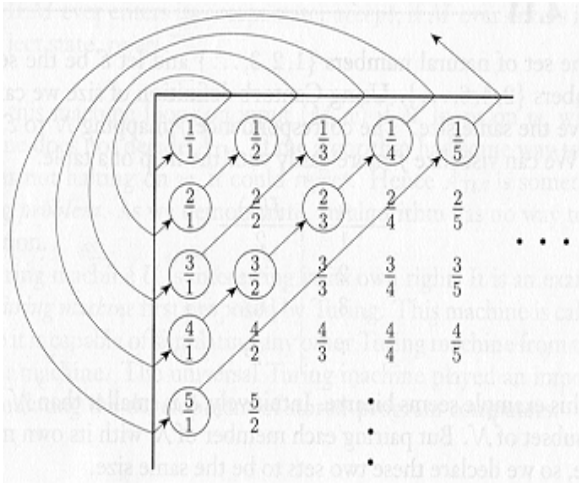
Another example

Let $\mathcal{Q}$ be the set of all the rational numbers, $\{\frac{m}{n} : m, n \in \mathcal{N}\}$.

Question: Which one contains more elements?

Again, intuitively, $|\mathcal{Q}| > |\mathcal{N}|$, since the former contains all the fractions, besides the natural numbers. Notice that for all $n, n = n/1$.

But, they are actually of the same size, because of the following 1-1 correspondence from $\mathcal{Q}$ to $\mathcal{N}$:



Could they be all the same?

Question: Are all the infinite sets of the same size?

Probably not. Otherwise, the world won't be as interesting... ☹

Definition: A set A is *countable* if A is either finite or $|A| = |\mathcal{N}|$.

A better question: Is it true all the infinite sets are countable?

A short answer: No! In fact, if the answer were positive, cantor's theory would not be interesting, at all.

A longer one: Let $\mathcal{C}$ be the set of all the real numbers, i.e., the numbers that have decimal representation, e.g., $3.1415926\cdots$ etc. Then

$$|\mathcal{C}| > |\mathcal{N}|.$$

Question: Why?

$\mathcal{C}$ is not countable.

Kick in from the back door again: Just assume $\mathcal{C}$ is countable.

By definition, since it is not finite, there exists a one-to-one and onto function, $f : \mathcal{N} \mapsto \mathcal{C}$, such that it will map every natural number to a unique real number, and, for every real number, there is a corresponding natural number.

In other words, all the real numbers can be listed, enumerated, as follows:

$$\begin{aligned}
f(1) &= x_{10}.\underline{x_{11}}x_{12}x_{13}\cdots \\
f(2) &= x_{20}.x_{21}\underline{x_{22}}x_{23}\cdots \\
f(3) &= x_{30}.x_{31}x_{32}\underline{x_{33}}\cdots \\
&\cdots \\
f(n) &= x_{n0}.x_{n1}x_{n2}x_{n3}\cdots\underline{x_{nn}}\cdots \\
&\cdots
\end{aligned}$$

Question: So what?

The punch line

We now construct a real number as follows:

$$y = 0.y_1 y_2 \cdots y_i \cdots$$

such that for all $i \in \mathcal{N}, y_i \neq f(i)_i$, the i^{th} digit of $f(i)$. In particular, $y_1 \neq f(1)_1, y_2 \neq f(2)_2, ..., y_i \neq f(i)_i, ...$

As y is obviously a real number, by definition of f, it is surjective, i.e., there must be a natural number m, such that

$$f(m) = y = y_{m0}.y_{m1}y_{m2}\cdots$$

However, by construction,

$f(m)_m \neq y_{m,m} = f(m)_m$, thus $f(m) \neq f(m)$, ☹ which is a contradiction. ☺ Thus, *such an* m *does not exit.* So, there does not exist such a one-to-one, and onto, function f, that would enumerate all the elements of $\mathcal{C}$.

As a result, $|\mathcal{C}| \neq |\mathcal{N}|$, i.e., $\mathcal{C}$ is not countable. Since $\mathcal{N} \subset \mathcal{C}$, $\mathcal{C} > \mathcal{N}$. □

How about an example?

If such a listing of all the real numbers is given as follows:

$$\begin{aligned} f(1) &= 0.\underline{2}5793 \\ f(2) &= 3.1\underline{4}92653589\cdots \\ f(3) &= 2.71\underline{8}281828\cdots \\ f(4) &= 0.577\underline{2}156649 \\ &\cdots \end{aligned}$$

Now, construct a real y as $0.3593654\cdots$. By the *onto* assumption made on f, there must exist m such that

$$f(m) = 0.3593\underline{6}54\cdots.$$

Assume $m = 6$. By definition of $f(m)$,

$$f(6)_6 \neq 5 = f(6)_6.$$

Indeed, there cannot be any such an m that $f(m) = y = 0.3593654\cdots$ for any listing f.

In other words, any such an f cannot be an "onto" function.

Nothing is new under the Sun...

From a logic perspective, the whole reasoning is still

$$A \rightarrow B \equiv \neg B \rightarrow \neg A.$$

Here A is $|\mathcal{C}| = |\mathcal{N}|$, and B is the existence of a one-to-one mapping from $\mathcal{N}$ to $\mathcal{C}$, based on Cantor's idea.

On the other hand, technically speaking, the catalyst is this *diagonalization technique,*

$$x_{1,1}x_{2,2}, \ldots, x_{n,n}, \ldots,$$

which plays a critical role in the derivation of this result, and others, as we will see, including the following one:

There exist languages that are not enumerable, since *there are more language (pigeons) than TMs (holes).*

Question: How could we show *that*?

Homework: Exercise 4.6

TMs are countable...

To show that the set of TMs is countable, we merely observe that the set of all strings of Σ^* for any finite alphabet Σ is countable.

With only finite number of strings of a given length, we can list all strings of length 0, followed by all strings of length 1, etc.. For example, let $\Sigma_0 = \{a, b\}$, its associated sequence is $\epsilon, a, b, aa, ab, ba, bb, aaa, aab, ..., bbb, ...$

A TM, $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, is a finite sequence of quintuples, $(q, a, b,\ p, \{L, R\})$, based on M,, and the collection of all the strings over such an alphabet is countable.

We now just cross over those strings that don't represent a TM, to get a listing similar to the one as shown on Page 20, we end up with a list of all the TMs.

Thus, the conclusion holds.

.... but languages are not.

For any Σ^*, let $\mathcal{B}$ be the collection of all the infinite sequences consisting of 0 and 1, where 1 indicates a string is present, 0 otherwise.

Now such a b_k represents the k^{th} string in Σ_0^*, $b_0 = 0 \cdots 00 \cdots$ represents ϵ, $b_4 = 000010 \cdots$, and $b_8 = 000000001 \cdots$, represents ab and aab, respectively (Cf. Page 26). Thus, a language is just a set of such binary sequences.

Assume $\mathcal{L}$, the set of all the languages, $\{L_k\}$, is countable, there must be a one-to-one mapping $f : \mathcal{N} \mapsto \mathcal{L}$, such that $f(k) = L_k, k \geq 1$.

$$
\begin{array}{rcl}
1 & \mapsto & L_1 = \{b_1, b_3, b_7\} \\
2 & \mapsto & L_2 = \{b_3, b_8\} \\
3 & \mapsto & L_3 = \{b_5, b_{10}, b_{12}\} \\
 & \cdots & \\
k & \mapsto & L_k = \{b_6, \cdots b_{k-2}, b_{150}\} \\
 & \cdots &
\end{array}
$$

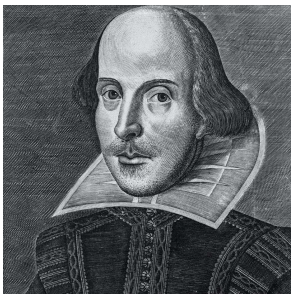
The punch line

Now we construct a language L, such that, for all $k, b_k \in L$ iff $b_k \notin L_k$. For example, $b_1 \notin L, b_2 \in L, b_3 \in L, b_k \in L \ldots\ldots$

Since L is a language, there must exist k_0, such that $L = f(k_0)$.

Question: is $b_{k_0} \in L_{k_0} = f(k_0) = L$?

Answer: Tough to say. If $b_{k_0} \in L_{k_0}$, it should not be in L; but if it is not, then it should. ☹



"*To be or not to be, that is the question.*"

Therefore, there does not exist such a function f. *Thus, the languages are not countable.*

The main result

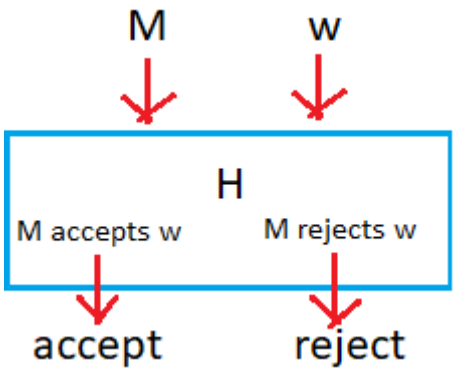
We are now ready to apply the diagonalization technique to show that A_{TM} is not decidable.

Theorem There does not exist a TM H, for any $\langle M, w \rangle$, H is able to tell whether *or not* M accepts w. That is, A_{TM} is not decidable.

Proof: Just assume it is decidable and H is its decider. Hence,

$$H(\langle M, w \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w, \\ \text{reject} & \text{otherwise.} \end{cases}$$

This machine looks like the following:



Now what?

We use H as a subroutine to construct a new TM, D, which determines what M does when the input to M is its own description, $\langle M \rangle$, which is a representation of itself.

For any input $\langle M \rangle$, D does the following:

0. Use the copier TM that you constructed as assigned on Page 10 of the TM notes to construct $\langle M, \langle M \rangle \rangle$.

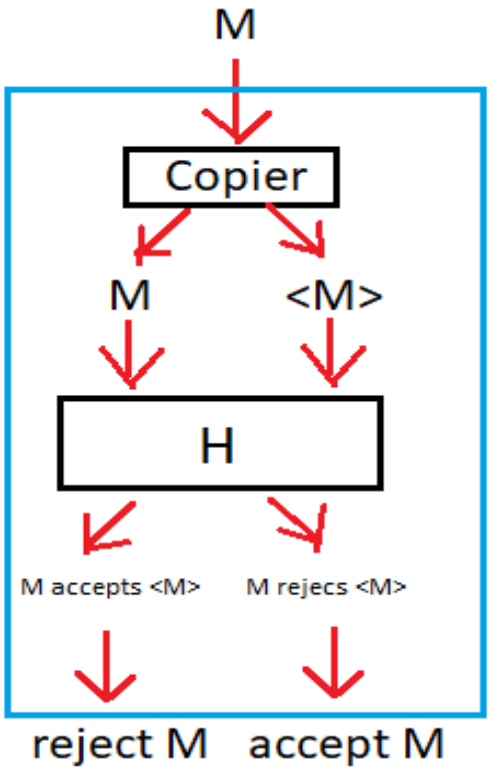
1. Run H on input $\langle M, \langle M \rangle \rangle$.

2. If H accepts $\langle M, \langle M \rangle \rangle$, i.e., M accepts $\langle M \rangle$, this machine, i.e., D, rejects $\langle M \rangle$;

3. If H rejects $\langle M, \langle M \rangle \rangle$, i.e., M rejects $\langle M \rangle$, D accepts $\langle M \rangle$.

I want to see...

It looks like the following:



This is what D would do with its input M.

$$D(M) = \begin{cases} \text{accept} & \text{if } M \text{ rejects } \langle M \rangle, \\ \text{reject} & \text{otherwise.} \end{cases}$$

Question: Now what?

The punch line...

Question: What happens when we run D with its own description, $\langle D \rangle$?

Answer: Replacing $\langle M \rangle$ with $\langle D \rangle$, we must have

$$D(\langle D \rangle) = \begin{cases} \text{accept} & \text{if } D \text{ rejects } \langle D \rangle, \\ \text{reject} & \text{if } D \text{ accepts } \langle D \rangle. \end{cases}$$

In other words,

D accepts $\langle D \rangle$ if and only if D rejects $\langle D \rangle$.

Clearly, there doesn't exist such a D, *because of this logical contradiction.* ☹

The non-existence of D ($\neg B$) implies that there could not exist such a decider H ($\neg A$) for the original Halting problem, either.

$$A \rightarrow B \equiv \neg B \rightarrow \neg A \ ☺$$

This ends the proof of this undecidability result of A_{TM}. □

Where is the beef?

The following assumes what M_i might do with $\langle M_i \rangle$, since TM is countable:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	Accept		Accept		
M_2	Accept	Accept	Accept	Accept	
M_3	Reject		Reject		
M_4	Accept	Accept			
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	

Assume that H exists, the following gives the values of $H(M_i, \langle M_i \rangle)$:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	Accept	**Reject**	Accept	**Reject**	
M_2	Accept	Accept	Accept	Accept	
M_3	Reject	**Reject**	Reject	**Reject**	
M_4	Accept	Accept	**Reject**	**Reject**	

Thus, H would fill all the holes for M_i.

Question: Then what?

The sneaky D...

The following shows what D would do and where the contradiction occurs:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$
M_1	Reject	reject	Accept	reject		Accept
M_2	Accept	Reject	Accept	Accept		Accept
M_3	reject	reject	Accept	reject		reject
M_4	Accept	Accept	reject	Accept		Accept
$\vdots$						
D	Accept	Accept	Reject	Reject		☹
$\vdots$						

By construction, D accepts, e.g., $\langle M_1 \rangle$, since M_1 rejects $\langle M_1 \rangle$, but rejects, e.g., $\langle M_3 \rangle$, since M_3 accepts $\langle M_3 \rangle$, ...

Question: What D would do with $\langle D \rangle$?

Again by construction, D should accept $\langle D \rangle$, if D rejects $\langle D \rangle$; and reject $\langle D \rangle$, if D accepts $\langle D \rangle$.

In other words, D *accepts* $\langle D \rangle$ *iff it rejects* $\langle D \rangle$.

We could not do it, no matter what. ☹

Thus, a contradiction occurs at "☹". ☺

Near the end

Recall that a language is *enumerable* if there exists a TM M that $L = L(M)$, the collection of all the words that M accepts (Cf. Page 15 of Chapter 4 notes).

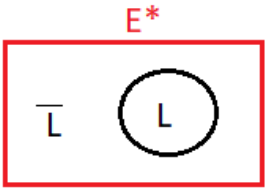
Procedurally, given a word $w \in \Sigma^*$, if $w \in L$, M will accept it; otherwise, M will not get into the accepting state: it either rejects w, or gets into an infinite loop.

We showed in Pages 26 through 28, that, since all the TMs are countable, but all the languages are not, at least one language is not associated with a TM. As a result, we concluded on Page 25 that some language is not TM enumerable.

We now show that $\overline{A_{\mathsf{TM}}}$ *is one of such languages.* $\overline{A_{\mathsf{TM}}}$ includes $\langle M, w \rangle$ such that M does not accept w.

A notion and a result

An enumerable language $\overline{L}$ is *co-enumerable* if it is the complement of an enumerable language L.



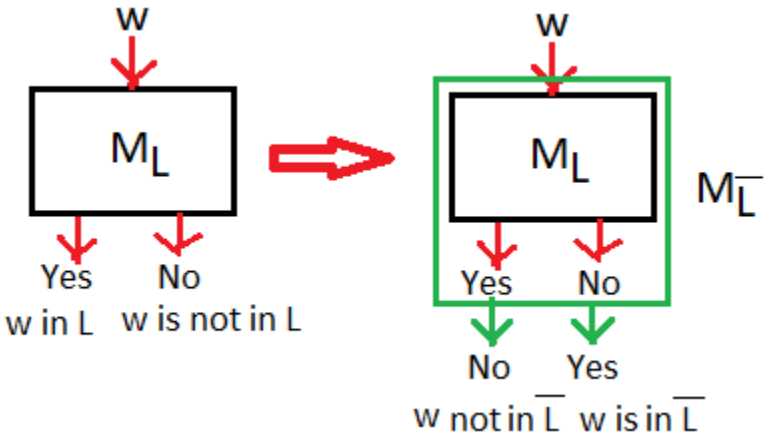
Theorem: A language is decidable if and only if it is both enumerable and co-enumerable.

Proof: Assume a language, L, is decidable, there is a TM M_L, that decides L, i.e., given an input w, it can tell either $w \in L$, or $w \notin L$.

In particular, M_L can tell if $w \in L$. Thus, L is enumerable.

The other piece...

We can then construct $M_{\overline{L}}$, based on M_L, thus, $\overline{L}$, the complement of a decidable language, L, is also decidable: $w \in \overline{L}$, iff $w \notin L$.

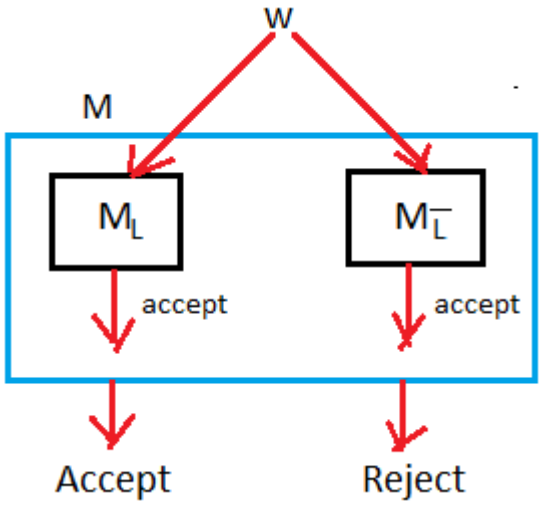


By the same token, $\overline{L}$, the complement of a decidable language, L, is enumerable.

Finally, by definition, L is co-enumerable, as it is the complement of an enumerable language $\overline{L}$.

We now show the other side, i.e., a language is decidable if it is both enumerable and co-enumerable.

By assumption, let M_L and $M_{\overline{L}}$ be TMs that accept an enumerable language, L, and its complement, $\overline{L}$, we construct the following TM, M, a decider for L.



Question: How would a math person interpret the above graph?

She might say the following ...

For any input w,

1. Run both M_L and $M_{\overline{L}}$ on input w in the "dovetail" style: i.e., each runs, alternatively, i (≥ 1) steps on w. (Cf. Page 31 of the Computability Chapter notes)

2. If M_L accepts w, accept; if $M_{\overline{L}}$ accepts w, reject.

As w either belongs to A or $\overline{A}$, either M_L or $M_{\overline{L}}$ will accept w. Thus, M is a decider for A.

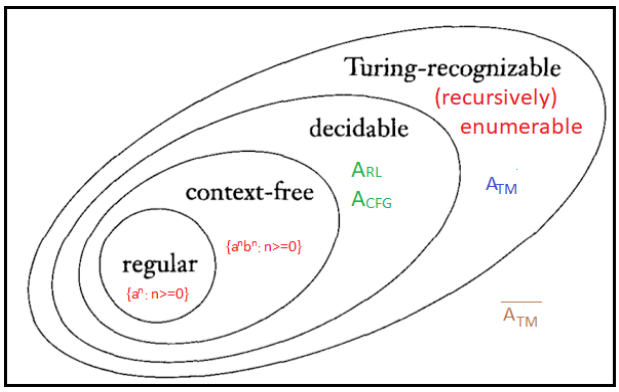
Corollary: $\overline{A_{\mathsf{TM}}}$ is not enumerable.

Proof: We know that A_{TM} is enumerable. (Cf. Page 14)

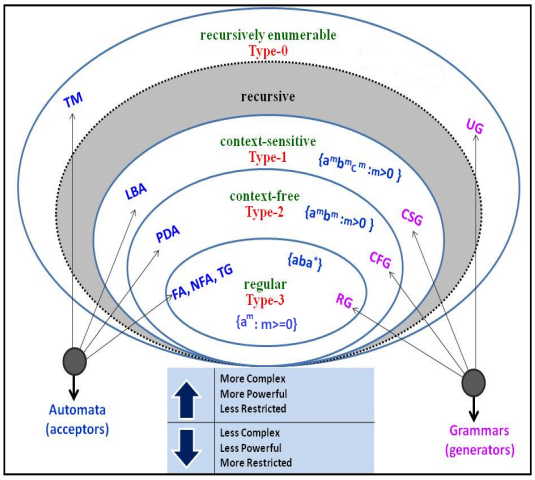
If $\overline{A_{\mathsf{TM}}}$ were also enumerable, A_{TM} would be decidable, by the previous result. This would contradict the main result on Page 29, i.e., A_{TM} is not decidable. □

The very end

This provides a concrete example outside the Turing-recognizable bubble on Page 12.



Let's beef up the one on Page 89 of the NPDA notes, as well. ☺



Wait a minute, A_{LBA} is decidable? Yes... . ☺