Algorithms and Computability

Lecture 9: Computability

Christian Schilling (christianms@cs.aau.dk)

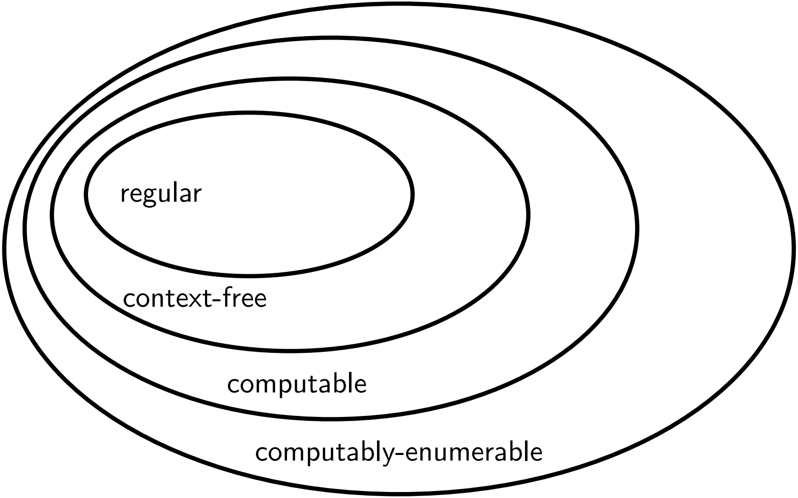
slides courtesy of Martin Zimmermann

We have seen

- the Church-Turing Thesis ("Everything that can be computed can be computed by a Turing machine"),
- multi-tape DTMs and
- nondeterministic TMs, and
- their equivalence: The same classes of languages are computably-enumerable (computable) by
  1. Deterministic one-tape TMs,
  2. Deterministic multi-tape TMs,
  3. Nondeterministic one-tape TMs, and
  4. Nondeterministic multi-tape TMs (not shown)

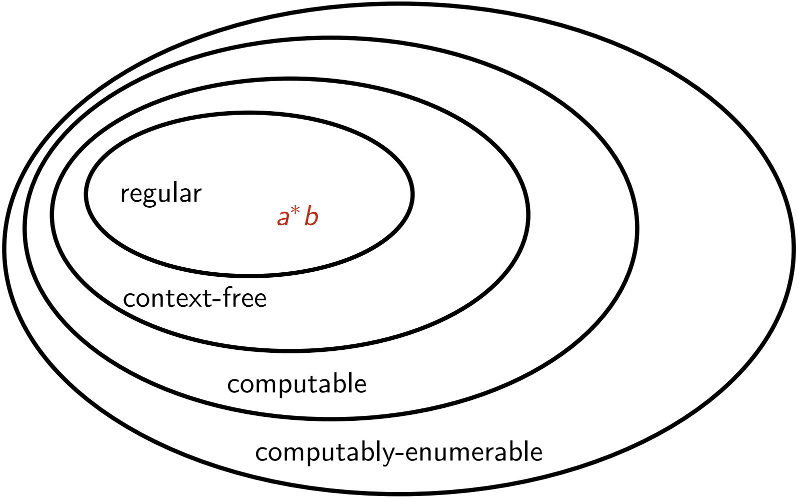
Today

Understand the structure of computable and computably-enumerable languages:



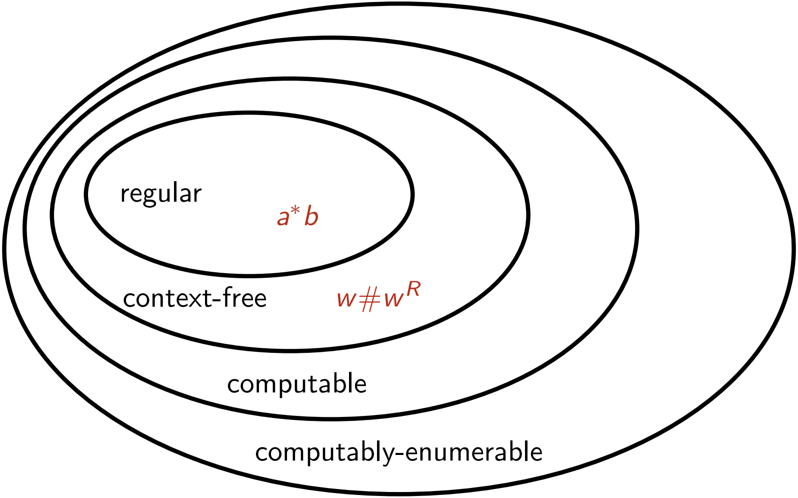
Today

Understand the structure of computable and computably-enumerable languages:



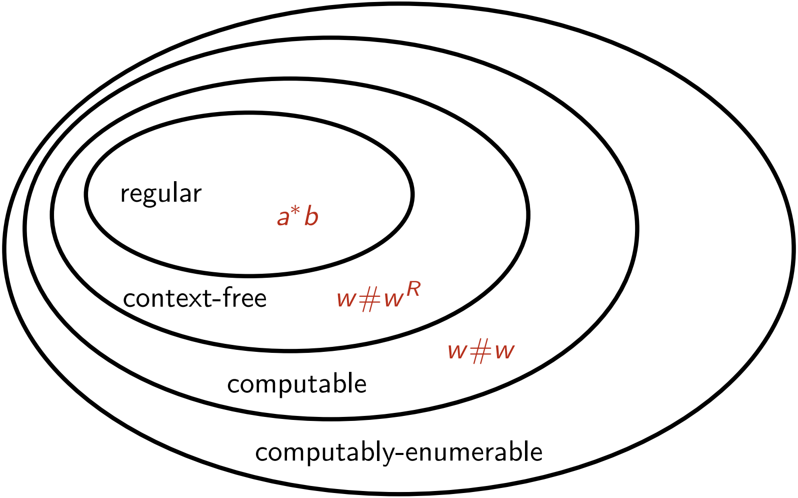
Today

Understand the structure of computable and computably-enumerable languages:



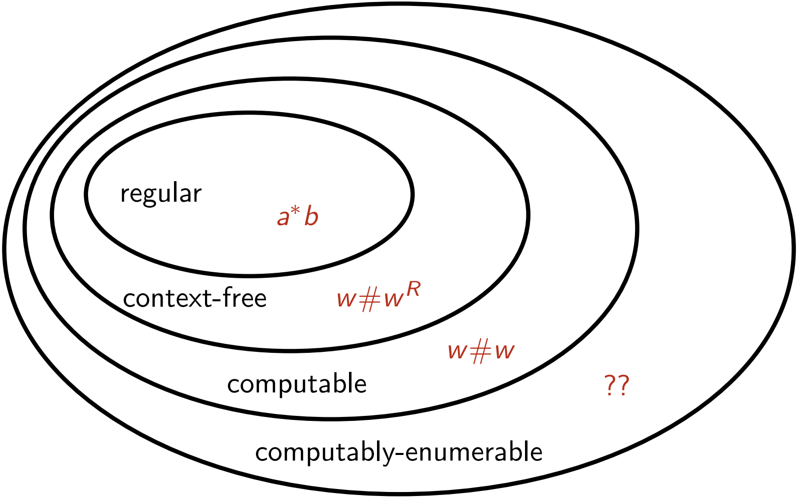
Today

Understand the structure of computable and computably-enumerable languages:



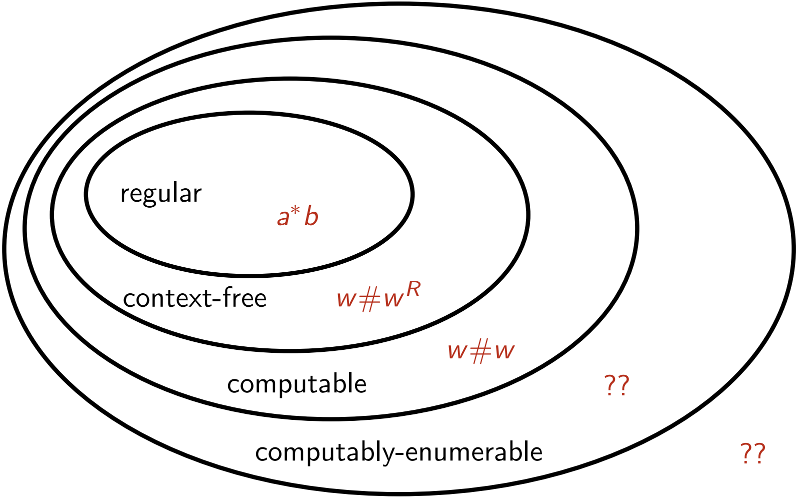
Today

Understand the structure of computable and computably-enumerable languages:



Today

Understand the structure of computable and computably-enumerable languages:



Agenda

1. Warm-up: Hotels and Barbers
2. The Halting Problem
3. Closure Properties

Hilbert's Grand Hotel

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest?

0	1	2	3	4	5	6	7	8	9

Hilbert's Grand Hotel

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest?

0	1	2	3	4	5	6	7	8	9	10	11

Hilbert's Grand Hotel

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

0	1	2	3	4	5	6	7	8	9	10	11

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	1	2	3	4	5	6	7	8	9	10	11
	0										

Hilbert's Grand Hotel

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	2	3	4	5	6	7	8	9	10	11

1

Hilbert's Grand Hotel

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	1	3	4	5	6	7	8	9	10	11
			2								

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	1	2	4	5	6	7	8	9	10	11
				3							

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	1	2	3	5	6	7	8	9	10	11

4

Hilbert's Grand Hotel

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	1	2	3	4	6	7	8	9	10	11
						5					

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	1	2	3	4	5	7	8	9	10	11
							6				

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	1	2	3	4	5	6	8	9	10	11
								7			

Hilbert's Grand Hotel

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	1	2	3	4	5	6	7	9	10	11
									8		

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	1	2	3	4	5	6	7	8	10	11
										9	

Hilbert's Grand Hotel

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	1	2	3	4	5	6	7	8	9	11

10

Hilbert's Grand Hotel

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	1	2	3	4	5	6	7	8	9	10

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	1	2	3	4	5	6	7	8	9	10

He can even accommodate infinitely many newly arriving guests in his hotel!

Hilbert's Grand Hotel

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

0	1	2	3	4	5	6	7	8	9	10	11

He can even accommodate infinitely many newly arriving guests in his hotel!

Hilbert's Grand Hotel

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	FREE	1	FREE	2	FREE	3	FREE	4	FREE	5

He can even accommodate infinitely many newly arriving guests in his hotel!

You have a hotel with ten rooms, all occupied. Can you accommodate a newly arriving guest? **No**

0	1	2	3	4	5	6	7	8	9

David Hilbert has a hotel with infinitely many rooms, all occupied. Can he accommodate a newly arriving guest? **Yes**

FREE	0	FREE	1	FREE	2	FREE	3	FREE	4	FREE	5

He can even accommodate infinitely many newly arriving guests in his hotel!

Infinite sets (and infinite hotels) behave quite differently from finite ones!

Are there Different Types of Infinities?

Question

Are there more subsets of natural numbers than natural numbers?

Are there Different Types of Infinities?

Question

Are there more subsets of natural numbers than natural numbers?

How to make sense of that question? There are both infinitely many natural numbers and infinitely many subsets of numbers. So, when do two sets have equal cardinality?

Are there Different Types of Infinities?

Question

Are there more subsets of natural numbers than natural numbers?

How to make sense of that question? There are both infinitely many natural numbers and infinitely many subsets of numbers. So, when do two sets have equal cardinality?

- For finite sets: if they have the same (finite) number of elements

$$\left\{ \begin{array}{c} 3, \\ 7, \\ 13 \end{array} \right\} \qquad\qquad \left\{ \begin{array}{c} a, \\ p, \\ u \end{array} \right\}$$

Are there Different Types of Infinities?

Question

Are there more subsets of natural numbers than natural numbers?

How to make sense of that question? There are both infinitely many natural numbers and infinitely many subsets of numbers. So, when do two sets have equal cardinality?

- For finite sets: if they have the same (finite) number of elements

$$\left\{\begin{array}{c}3,\\ 7,\\ 13\end{array}\right\}\begin{array}{c}\longrightarrow\\ \longrightarrow\\ \longrightarrow\end{array}\left\{\begin{array}{c}a,\\ p,\\ u\end{array}\right\}$$

- Equivalently: there is a bijection between the sets

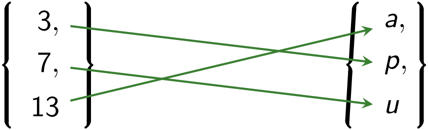
Are there Different Types of Infinities?

Question

Are there more subsets of natural numbers than natural numbers?

How to make sense of that question? There are both infinitely many natural numbers and infinitely many subsets of numbers. So, when do two sets have equal cardinality?

- For finite sets: if they have the same (finite) number of elements



- Equivalently: there is a bijection between the sets

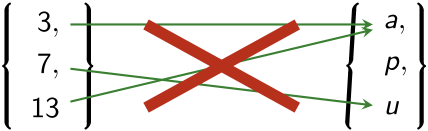
Are there Different Types of Infinities?

Question

Are there more subsets of natural numbers than natural numbers?

How to make sense of that question? There are both infinitely many natural numbers and infinitely many subsets of numbers. So, when do two sets have equal cardinality?

- For finite sets: if they have the same (finite) number of elements



- Equivalently: there is a bijection between the sets

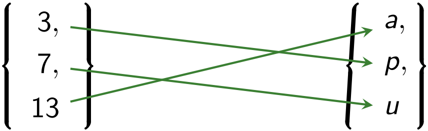
Are there Different Types of Infinities?

Question

Are there more subsets of natural numbers than natural numbers?

How to make sense of that question? There are both infinitely many natural numbers and infinitely many subsets of numbers. So, when do two sets have equal cardinality?

- For finite sets: if they have the same (finite) number of elements



- Equivalently: there is a bijection between the sets

We use the same approach for infinite sets

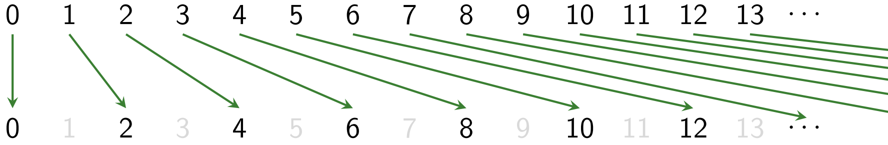
Unintuitive Consequences

“Two sets have the same cardinality if there is a bijection between them”

Unintuitive Consequences

"Two sets have the same cardinality if there is a bijection between them"

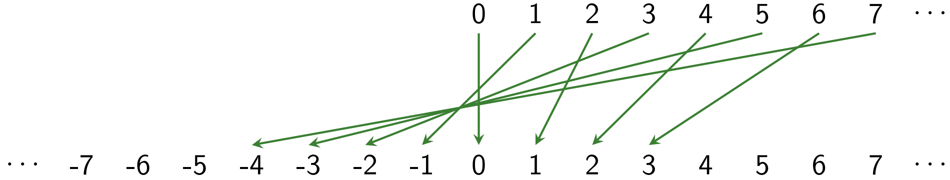
- So, the natural numbers and the even natural numbers have the same cardinality



Unintuitive Consequences

"Two sets have the same cardinality if there is a bijection between them"

- So, the natural numbers and the even natural numbers have the same cardinality
- And the natural numbers and the integers have the same cardinality



Unintuitive Consequences

"Two sets have the same cardinality if there is a bijection between them"

- So, the natural numbers and the even natural numbers have the same cardinality
- And the natural numbers and the integers have the same cardinality
- The natural numbers and Σ^* also have the same cardinality
- The latter result also implies that even the natural numbers and the rational numbers have the same cardinality
- But Cantor's diagonalization argument shows that the cardinality of the reals is strictly larger than that of the natural numbers

Back to our Question

- Assume the natural numbers and the subsets of the natural numbers have the same cardinality
- Then, there is a bijection f between the natural numbers and the subsets of the natural numbers
- So, the list $f(0), f(1), f(2), \ldots$ contains all subsets of the natural numbers

Back to our Question

- We can arrange the list $f(0), f(1), f(2), \ldots$ in an infinite table such that row i represents the subset $f(i)$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	$\cdots$
$f(0)$	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	$\cdots$
$f(1)$	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	$\cdots$
$f(2)$	✗	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✓	✗	✓	$\cdots$
$f(3)$	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	$\cdots$
$f(4)$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	$\cdots$
$f(5)$	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	$\cdots$
$f(6)$	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	$\cdots$
$f(7)$	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Back to our Question

- We can arrange the list $f(0), f(1), f(2), \ldots$ in an infinite table such that row i represents the subset $f(i)$
- Consider the set $\overline{D} = \{3, 5, 7, \ldots\}$ obtained by flipping the diagonal

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	$\cdots$
$f(0)$	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	$\cdots$
$f(1)$	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	$\cdots$
$f(2)$	✗	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✓	✗	✓	$\cdots$
$f(3)$	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	$\cdots$
$f(4)$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	$\cdots$
$f(5)$	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	$\cdots$
$f(6)$	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	$\cdots$
$f(7)$	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- We can arrange the list $f(0), f(1), f(2), \ldots$ in an infinite table such that row i represents the subset $f(i)$
- Consider the set $\overline{D} = \{3, 5, 7, \ldots\}$ obtained by flipping the diagonal
- It differs from every set $f(i)$ in the membership of i

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	$\cdots$
$f(0)$	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	$\cdots$
$f(1)$	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	$\cdots$
$f(2)$	✗	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✓	✗	✓	$\cdots$
$f(3)$	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	$\cdots$
$f(4)$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	$\cdots$
$f(5)$	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	$\cdots$
$f(6)$	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	$\cdots$
$f(7)$	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Back to our Question

- We can arrange the list $f(0), f(1), f(2), \ldots$ in an infinite table such that row i represents the subset $f(i)$
- Consider the set $\overline{D} = \{3, 5, 7, \ldots\}$ obtained by flipping the diagonal
- It differs from every set $f(i)$ in the membership of i
- Thus, $\overline{D}$ is not in the list $f(0), f(1), f(2), \ldots$

Back to our Question

- We can arrange the list $f(0), f(1), f(2), \ldots$ in an infinite table such that row i represents the subset $f(i)$
- Consider the set $\overline{D} = \{3, 5, 7, \ldots\}$ obtained by flipping the diagonal
- It differs from every set $f(i)$ in the membership of i
- Thus, $\overline{D}$ is not in the list $f(0), f(1), f(2), \ldots$
- This is a contradiction to our assumption that the list contains all subsets, i.e., our assumption must be wrong: there is no bijection between the natural numbers and the subsets of natural numbers

Back to our Question

- We can arrange the list $f(0), f(1), f(2), \ldots$ in an infinite table such that row i represents the subset $f(i)$
- Consider the set $\overline{D} = \{3, 5, 7, \ldots\}$ obtained by flipping the diagonal
- It differs from every set $f(i)$ in the membership of i
- Thus, $\overline{D}$ is not in the list $f(0), f(1), f(2), \ldots$
- This is a contradiction to our assumption that the list contains all subsets, i.e., our assumption must be wrong: there is no bijection between the natural numbers and the subsets of natural numbers

This process, due to Georg Cantor, is called **diagonalization** and has many applications in set theory, logic, and computability

Barber Paradox

Suppose a town's male barber shaves precisely every man in town who does not shave himself.
Who shaves the barber?

Barber Paradox

Suppose a town's male barber shaves precisely every man in town who does not shave himself.
Who shaves the barber?

- If the barber shaves himself, then he is not shaved by the barber. So, he does not shave himself

Barber Paradox

Suppose a town's male barber shaves precisely every man in town who does not shave himself.
Who shaves the barber?

- If the barber shaves himself, then he is not shaved by the barber. So, he does not shave himself
- If the barber does not shave himself, then he is shaved by the barber. So, he shaves himself

Barber Paradox

Suppose a town's male barber shaves precisely every man in town who does not shave himself.
Who shaves the barber?

- If the barber shaves himself, then he is not shaved by the barber. So, he does not shave himself
- If the barber does not shave himself, then he is shaved by the barber. So, he shaves himself

Both cases lead to a contradiction

Barber Paradox

Suppose a town's male barber shaves precisely every man in town who does not shave himself.
Who shaves the barber?

- If the barber shaves himself, then he is not shaved by the barber. So, he does not shave himself
- If the barber does not shave himself, then he is shaved by the barber. So, he shaves himself

Both cases lead to a contradiction

The paradox relies on **self-reference**, i.e., "does the barber shave *himself*?"

Barber Paradox

Suppose a town's male barber shaves precisely every man in town who does not shave himself.
Who shaves the barber?

- If the barber shaves himself, then he is not shaved by the barber. So, he does not shave himself
- If the barber does not shave himself, then he is shaved by the barber. So, he shaves himself

Both cases lead to a contradiction

The paradox relies on **self-reference**, i.e., "does the barber shave *himself*?"

Can we apply self-reference to Turing machines?

Quiz 1

The number 0 is interesting, as $n + 0 = n$ for all n. The number 1 is interesting, as $n \cdot 1 = n$ for all n. The number 2 is interesting, as it is the only even prime number. The number 3 is interesting, as it is the only number n that satisfies $n = \sum_{0 \leq j < n} j$

Is there an uninteresting number?

Quiz 1

The number 0 is interesting, as $n + 0 = n$ for all n. The number 1 is interesting, as $n \cdot 1 = n$ for all n. The number 2 is interesting, as it is the only even prime number. The number 3 is interesting, as it is the only number n that satisfies $n = \sum_{0 \leq j < n} j$

Is there an uninteresting number?

This is another paradox
If there was an uninteresting number, that fact itself would make it interesting

More paradoxes:
`https://en.wikipedia.org/wiki/List_of_paradoxes`

Agenda

1. Warm-up: Hotels and Barbers
2. **The Halting Problem**
3. Closure Properties

Encoding of Turing Machines

Let $M = (Q, \Sigma, \Gamma, s, t, r, \delta)$ be a DTM. We assume without loss of generality that $Q = \{1, 11, \ldots, 1^{|Q|}\}$ and $\Sigma = \{0, 1\}$.
Also, let $rep_\Gamma \colon \Gamma \to \{1, 11, \ldots, 1^{|\Gamma|}\}$ be an encoding of Γ such that $rep_\Gamma(0) = 1$, $rep_\Gamma(1) = 11$, and $rep_\Gamma(\text{␣}) = 111$

Encoding of Turing Machines

Let $M = (Q, \Sigma, \Gamma, s, t, r, \delta)$ be a DTM. We assume without loss of generality that $Q = \{1, 11, \ldots, 1^{|Q|}\}$ and $\Sigma = \{0, 1\}$.
Also, let $rep_\Gamma : \Gamma \to \{1, 11, \ldots, 1^{|\Gamma|}\}$ be an encoding of Γ such that $rep_\Gamma(0) = 1$, $rep_\Gamma(1) = 11$, and $rep_\Gamma(\textvisiblespace) = 111$

Then, M is encoded by the word $\ulcorner M \urcorner$ over $\{0, 1\}$ defined as follows:

$$1^{|Q|}\,0\,1^{|\Gamma|}\,0\,s\,0\,t\,0\,r\,0\,w_\delta$$

where w_δ is the list of encodings of transitions.
Each $\delta(q, b) = (p, a, d)$ is encoded by

$$q\,0\,rep_\Gamma(b)\,0\,p\,0\,rep_\Gamma(a)\,0\,dir(d)\,0$$

where $dir(-1) = 1$ and $dir(+1) = 11$

Universal Turing Machines

- So, (the encoding of) a DTM can be the input for a DTM

Universal Turing Machines

- So, (the encoding of) a DTM can be the input for a DTM
- We can even construct a universal DTM that takes as input
  1. the encoding $\ulcorner M \urcorner$ of a DTM M, and
  2. an input w for M,

  and then simulates the run of M on w
- We do not show that here

Universal Turing Machines

- So, (the encoding of) a DTM can be the input for a DTM
- We can even construct a universal DTM that takes as input
  1. the encoding $\ulcorner M \urcorner$ of a DTM M, and
  2. an input w for M,

  and then simulates the run of M on w
- We do not show that here

Note

Basis of modern computer architecture: program and input are both data stored in memory (von Neumann architecture)

The Halting Problem

- So, (the encoding of) a DTM can be the input for a DTM
- Thus, we can formulate decision problems about DTMs!

The Halting Problem

- So, (the encoding of) a DTM can be the input for a DTM
- Thus, we can formulate decision problems about DTMs!
- A particular interesting (both practically and theoretically) problem is the **halting problem**:

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w \rangle \mid M \text{ is a DTM that halts on input } w\}$$

- Here, $\langle \cdot, \cdot \rangle$ is a function that encodes two words x, y over $\{0, 1\}$ by a single word $\langle x, y \rangle$ over $\{0, 1\}$. See "Encoding pairs" in "Computability and Complexity" (page 89) for details

The Halting Problem

- So, (the encoding of) a DTM can be the input for a DTM
- Thus, we can formulate decision problems about DTMs!
- A particular interesting (both practically and theoretically) problem is the **halting problem**:

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w\rangle \mid M \text{ is a DTM that halts on input } w\}$$

- Here, $\langle \cdot, \cdot \rangle$ is a function that encodes two words x, y over $\{0, 1\}$ by a single word $\langle x, y\rangle$ over $\{0, 1\}$. See "Encoding pairs" in "Computability and Complexity" (page 89) for details

Theorem (Turing 1936)

The halting problem is not computable

Intuition

- List behavior of all halting DTMs M_i on all inputs w_j in matrix

- List behavior of all halting DTMs M_i on all inputs w_j in matrix

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	$\cdots$
M_0	acc	rej	acc	rej	acc	rej	acc	rej	acc	rej	$\cdots$
M_1	rej	acc	rej	acc	rej	acc	rej	acc	rej	acc	$\cdots$
M_2	rej	rej	acc	acc	rej	acc	rej	acc	rej	rej	$\cdots$
M_3	rej	rej	rej	rej	rej	rej	rej	rej	rej	rej	$\cdots$
M_4	acc	acc	acc	acc	acc	acc	acc	acc	rej	acc	$\cdots$
M_5	rej	acc	acc	rej	acc	rej	rej	rej	acc	rej	$\cdots$
M_6	acc	rej	rej	acc	rej	rej	acc	rej	rej	acc	$\cdots$
M_7	acc	rej	rej	acc	rej	rej	rej	rej	rej	acc	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Intuition

- List behavior of all halting DTMs M_i on all inputs w_j in matrix
- Flipping the diagonal gives language $\{w_3, w_5, w_7, \ldots\}$ that is different from every computable language

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	$\cdots$
M_0	acc	rej	acc	rej	acc	rej	acc	rej	acc	rej	$\cdots$
M_1	rej	acc	rej	acc	rej	acc	rej	acc	rej	acc	$\cdots$
M_2	rej	rej	acc	acc	rej	acc	rej	acc	rej	rej	$\cdots$
M_3	rej	rej	rej	rej	rej	rej	rej	rej	rej	rej	$\cdots$
M_4	acc	acc	acc	acc	acc	acc	acc	acc	rej	acc	$\cdots$
M_5	rej	acc	acc	rej	acc	rej	rej	rej	acc	rej	$\cdots$
M_6	acc	rej	rej	acc	rej	rej	acc	rej	rej	acc	$\cdots$
M_7	acc	rej	rej	acc	rej	rej	rej	rej	rej	acc	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Proof

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w \rangle \mid M \text{ is a DTM that halts on input } w\}$$

- Assume HP is computable by some halting DTM H

Proof

$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w\rangle \mid M \text{ is a DTM that halts on input } w\}$

- Assume HP is computable by some halting DTM H
- Consider the following DTM B (built from H):
  1. Given an input w, simulate H on the input $\langle w, w\rangle$
  2. If H accepts, then go into an infinite loop
  3. If H rejects, then accept

Proof

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w \rangle \mid M \text{ is a DTM that halts on input } w\}$$

- Assume HP is computable by some halting DTM H
- Consider the following DTM B (built from H):
  1. Given an input w, simulate H on the input $\langle w, w \rangle$
  2. If H accepts, then go into an infinite loop
  3. If H rejects, then accept

Does B halt when applied to the input $\ulcorner B \urcorner$?

Proof

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w\rangle \mid M \text{ is a DTM that halts on input } w\}$$

- Assume HP is computable by some halting DTM H
- Consider the following DTM B (built from H):
  1. Given an input w, simulate H on the input $\langle w, w\rangle$
  2. If H accepts, then go into an infinite loop
  3. If H rejects, then accept

Does B halt when applied to the input $\ulcorner B \urcorner$?

- Case 1: B halts on input $\ulcorner B \urcorner$
  - Then, H accepts $\langle \ulcorner B \urcorner, \ulcorner B \urcorner\rangle$ by definition of HP
  - Hence, B does not halt on input $\ulcorner B \urcorner$

Proof

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w\rangle \mid M \text{ is a DTM that halts on input } w\}$$

- Assume HP is computable by some halting DTM H
- Consider the following DTM B (built from H):
  1. Given an input w, simulate H on the input $\langle w, w\rangle$
  2. If H accepts, then go into an infinite loop
  3. If H rejects, then accept

Does B halt when applied to the input $\ulcorner B \urcorner$?

- Case 1: B halts on input $\ulcorner B \urcorner$
  - Then, H accepts $\langle \ulcorner B \urcorner, \ulcorner B \urcorner\rangle$ by definition of HP
  - Hence, B does not halt on input $\ulcorner B \urcorner$
- Case 2: B does not halt on input $\ulcorner B \urcorner$
  - Then, H rejects $\langle \ulcorner B \urcorner, \ulcorner B \urcorner\rangle$ by definition of HP
  - Hence, B halts on input $\ulcorner B \urcorner$

Proof

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w \rangle \mid M \text{ is a DTM that halts on input } w\}$$

- Assume HP is computable by some halting DTM H
- Consider the following DTM B (built from H):
  1. Given an input w, simulate H on the input $\langle w, w \rangle$
  2. If H accepts, then go into an infinite loop
  3. If H rejects, then accept

Does B halt when applied to the input $\ulcorner B \urcorner$?

- Case 1: B halts on input $\ulcorner B \urcorner$
  - Then, H accepts $\langle \ulcorner B \urcorner, \ulcorner B \urcorner \rangle$ by definition of HP
  - Hence, B does not halt on input $\ulcorner B \urcorner$
- Case 2: B does not halt on input $\ulcorner B \urcorner$
  - Then, H rejects $\langle \ulcorner B \urcorner, \ulcorner B \urcorner \rangle$ by definition of HP
  - Hence, B halts on input $\ulcorner B \urcorner$

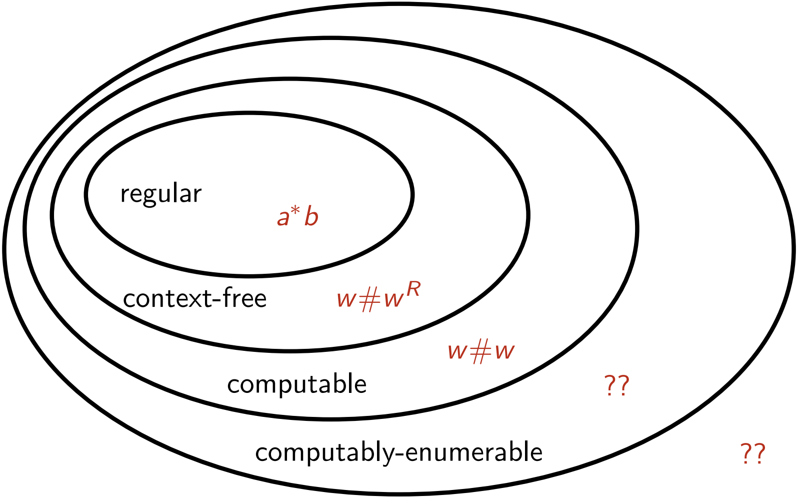
Both cases lead to a contradiction, so H cannot exist
Hence, the halting problem HP is not computable

What about Computably-enumerability?

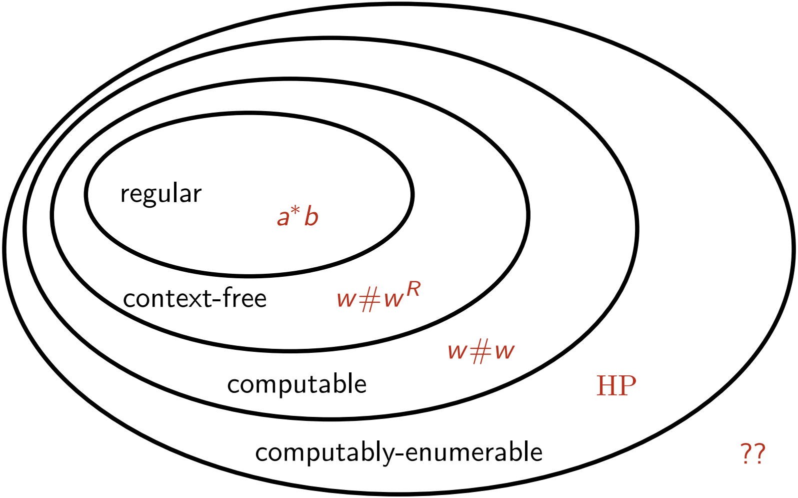
Is the halting problem at least computably-enumerable?

Exercise 5: Yes, it is!

What Have We Achieved?



What Have We Achieved?



Agenda

1. Warm-up: Hotels and Barbers
2. The Halting Problem
3. **Closure Properties**

Reminder: Operations on Languages

Let L_1, L_2 be two languages over Σ

- union:

$$L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ or } w \in L_2\}$$

- intersection:

$$L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ and } w \in L_2\}$$

- complement (w.r.t. Σ^*):

$$\overline{L_1} = \{w \in \Sigma^* \mid w \notin L_1\}$$

- concatenation:

$$L_1 \cdot L_2 = \{w \in \Sigma^* \mid w = w_1 w_2 \text{ with } w_1 \in L_1 \text{ and } w_2 \in L_2\}$$

- Kleene star (iteration):

$$(L_1)^* = \{w \in \Sigma^* \mid w = w_1 w_2 \cdots w_k \text{ for some } k \geq 0 \text{ and } w_i \in L_1 \text{ for all } i \in \{1, 2, \dots, k\}\}$$

Why Care about Closure Properties?

There are two main applications of closure properties:

1. Show that a language is in some class of languages

Why Care about Closure Properties?

There are two main applications of closure properties:

1. Show that a language is in some class of languages

Example

- Regular languages are closed under intersection
- $L_s = \{w \in \mathbb{B}^* \mid w \text{ starts with a } 1\}$ is regular
- $L_e = \{w \in \mathbb{B}^* \mid w \text{ ends with a } 1\}$ is regular
- So, $\{w \in \mathbb{B}^* \mid w \text{ starts and ends with a } 1\} = L_s \cap L_e$ is regular as well

Why Care about Closure Properties?

There are two main applications of closure properties:

1. Show that a language is in some class of languages
2. Show that a language is **not** in some class of languages

Why Care about Closure Properties?

There are two main applications of closure properties:

1. Show that a language is in some class of languages
2. Show that a language is **not** in some class of languages

Example

- Regular languages are closed under intersection and complement
- $L_{prs} = \{x\#y \mid x, y \in \mathbb{B}^*\}$ is regular

Why Care about Closure Properties?

There are two main applications of closure properties:

1. Show that a language is in some class of languages
2. Show that a language is **not** in some class of languages

Example

- Regular languages are closed under intersection and complement
- $L_{prs} = \{x\#y \mid x, y \in \mathbb{B}^*\}$ is regular
- Assume $L_{ne} = \{x\#y \mid x, y \in \mathbb{B}^*$ such that $x \neq y\}$ is regular

Why Care about Closure Properties?

There are two main applications of closure properties:

1. Show that a language is in some class of languages
2. Show that a language is **not** in some class of languages

Example

- Regular languages are closed under intersection and complement
- $L_{prs} = \{x\#y \mid x, y \in \mathbb{B}^*\}$ is regular
- Assume $L_{ne} = \{x\#y \mid x, y \in \mathbb{B}^* \text{ such that } x \neq y\}$ is regular
- Then, $L = \overline{L_{ne}} \cap L_{prs}$ is regular as well

Why Care about Closure Properties?

There are two main applications of closure properties:

1. Show that a language is in some class of languages
2. Show that a language is **not** in some class of languages

Example

- Regular languages are closed under intersection and complement
- $L_{prs} = \{x\#y \mid x, y \in \mathbb{B}^*\}$ is regular
- Assume $L_{ne} = \{x\#y \mid x, y \in \mathbb{B}^* \text{ such that } x \neq y\}$ is regular
- Then, $L = \overline{L_{ne}} \cap L_{prs}$ is regular as well
- But $L = \{w\#w \mid w \in \mathbb{B}^*\}$, which we know is not regular
- Contradiction. So, our assumption is wrong and L_{ne} is not regular

Intersections

Given one-tape DTMs M_1 for $L_1 \subseteq \Sigma^*$ and M_2 for $L_2 \subseteq \Sigma^*$, we construct a two-tape DTM for $L_1 \cap L_2$:

On input w:

1. Copy w on second tape, place both heads at first letter of w
2. Do one step of M_1 on first tape and one step of M_2 on second tape (if one halts, only simulate the other)
3. If both M_1 and M_2 accept, then accept as well
4. If both M_1 and M_2 have halted, then reject
5. Go to step 2

Intersections

Given one-tape DTMs M_1 for $L_1 \subseteq \Sigma^*$ and M_2 for $L_2 \subseteq \Sigma^*$, we construct a two-tape DTM for $L_1 \cap L_2$:

On input w:

1. Copy w on second tape, place both heads at first letter of w
2. Do one step of M_1 on first tape and one step of M_2 on second tape (if one halts, only simulate the other)
3. If both M_1 and M_2 accept, then accept as well
4. If both M_1 and M_2 have halted, then reject
5. Go to step 2

This DTM accepts if and only if the input is in L_1 and in L_2

Intersections

Given one-tape DTMs M_1 for $L_1 \subseteq \Sigma^*$ and M_2 for $L_2 \subseteq \Sigma^*$, we construct a two-tape DTM for $L_1 \cap L_2$:

On input w:

1. Copy w on second tape, place both heads at first letter of w
2. Do one step of M_1 on first tape and one step of M_2 on second tape (if one halts, only simulate the other)
3. If both M_1 and M_2 accept, then accept as well
4. If both M_1 and M_2 have halted, then reject
5. Go to step 2

This DTM accepts if and only if the input is in L_1 and in L_2

Note: If both M_1 and M_2 are halting, then the constructed DTM for $L_1 \cap L_2$ is also halting

Unions, Approach 1

Given one-tape DTMs M_1 for $L_1 \subseteq \Sigma^*$ and M_2 for $L_2 \subseteq \Sigma^*$, we construct a two-tape DTM for $L_1 \cup L_2$:

On input w:

1. Copy w on second tape, place both heads at first letter of w
2. Do one step of M_1 on first tape and one step of M_2 on second tape (if one has halted only simulate the other)
3. If one of M_1 and M_2 accepts, then accept as well
4. If both M_1 and M_2 have halted, then reject
5. Go to step 2

Unions, Approach 1

Given one-tape DTMs M_1 for $L_1 \subseteq \Sigma^*$ and M_2 for $L_2 \subseteq \Sigma^*$, we construct a two-tape DTM for $L_1 \cup L_2$:

On input w:

1. Copy w on second tape, place both heads at first letter of w
2. Do one step of M_1 on first tape and one step of M_2 on second tape (if one has halted only simulate the other)
3. If one of M_1 and M_2 accepts, then accept as well
4. If both M_1 and M_2 have halted, then reject
5. Go to step 2

This DTM accepts if and only if the input is in L_1 or in L_2

Unions, Approach 1

Given one-tape DTMs M_1 for $L_1 \subseteq \Sigma^*$ and M_2 for $L_2 \subseteq \Sigma^*$, we construct a two-tape DTM for $L_1 \cup L_2$:

On input w:

1. Copy w on second tape, place both heads at first letter of w
2. Do one step of M_1 on first tape and one step of M_2 on second tape (if one has halted only simulate the other)
3. If one of M_1 and M_2 accepts, then accept as well
4. If both M_1 and M_2 have halted, then reject
5. Go to step 2

This DTM accepts if and only if the input is in L_1 or in L_2

Note: If both M_1 and M_2 are halting, then the constructed DTM for $L_1 \cup L_2$ is also halting

Unions, Approach 2

Given one-tape DTMs M_1 for $L_1 \subseteq \Sigma^*$ and M_2 for $L_2 \subseteq \Sigma^*$, we construct a nondeterministic one-tape DTM for $L_1 \cup L_2$:

On input w:

1. Nondeterministically choose $i \in \{1, 2\}$
2. Run M_i on w
3. If M_i accepts, accept as well
4. If M_i rejects, reject as well

Unions, Approach 2

Given one-tape DTMs M_1 for $L_1 \subseteq \Sigma^*$ and M_2 for $L_2 \subseteq \Sigma^*$, we construct a nondeterministic one-tape DTM for $L_1 \cup L_2$:

On input w:

1. Nondeterministically choose $i \in \{1, 2\}$
2. Run M_i on w
3. If M_i accepts, accept as well
4. If M_i rejects, reject as well

This NTM accepts if and only if the input is in L_1 or in L_2

Unions, Approach 2

Given one-tape DTMs M_1 for $L_1 \subseteq \Sigma^*$ and M_2 for $L_2 \subseteq \Sigma^*$, we construct a nondeterministic one-tape DTM for $L_1 \cup L_2$:

On input w:

1. Nondeterministically choose $i \in \{1, 2\}$
2. Run M_i on w
3. If M_i accepts, accept as well
4. If M_i rejects, reject as well

This NTM accepts if and only if the input is in L_1 or in L_2

Note: If both M_1 and M_2 are halting, then the constructed NTM for $L_1 \cup L_2$ is also halting

Complements of Computable Languages

Given a deterministic one-tape **halting** DTM M for $L \subseteq \Sigma^*$, we construct a deterministic halting DTM for $\overline{L}$:

On input w:

1. Run M on w
2. If M accepts, then reject
3. If M rejects, then accept

Complements of Computable Languages

Given a deterministic one-tape **halting** DTM M for $L \subseteq \Sigma^*$, we construct a deterministic halting DTM for $\overline{L}$:

On input w:

1. Run M on w
2. If M accepts, then reject
3. If M rejects, then accept

This DTM is halting (as M is halting) and accepts w if and only if M rejects w

Complements of Computable Languages

Given a deterministic one-tape **halting** DTM M for $L \subseteq \Sigma^*$, we construct a deterministic halting DTM for $\overline{L}$:

On input w:

1. Run M on w
2. If M accepts, then reject
3. If M rejects, then accept

This DTM is halting (as M is halting) and accepts w if and only if M rejects w

Does the same construction work for computably-enumerable languages?

Closure Properties

Theorem

1. *The computable languages are closed under union, intersection, complement, concatenation, and iteration*
2. *The computably-enumerable languages are closed under union, intersection, concatenation, and iteration, but* **not** *under complement*

Closure Properties

Theorem

1. *The computable languages are closed under union, intersection, complement, concatenation, and iteration*
2. *The computably-enumerable languages are closed under union, intersection, concatenation, and iteration, but* **not** *under complement*

- We sketched some of the proofs here, but not all
- Some more are on the exercise sheet
- The others are similar (and can be found in the literature)

Conclusion

We have seen

- Diagonalization
- Non-computability of the halting problem
- Closure properties of computable and computably-enumerable languages

Reading

In "Computability and Complexity":

- Section 2.3 on universal Turing machines
- Section 2.4 on non-computable problems
- Section 2.5 on closure properties