Teoria da Computação
2018/2

Álvaro Moreira
afmoreira@inf.ufrgs.br

Instituto de Informática
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brasil
http://www.inf.ufrgs.br

Contents

Contents

Decidable Languages

In previous class we saw some examples of languages that are decidable:

- $A = \{0^{2^n} \mid n \geq 0\}$
- $D_1 = \{\langle p \rangle \mid p \text{ is a polynomial over } x \text{ with an integral root}\}$

Chapter 3 of Sipser's book has also TMs that decide these languages:

- $E = \{\sharp x_1 \sharp x_2 \sharp \ldots \sharp x_k \mid \text{each } x_i \in \{0,1\}^*, \text{ and } x_i \neq x_j \text{ for each } i \neq j\}$
- $A = \{\langle G \rangle \mid G \text{ is a connected and undirected graph}\}$

(obs.: the TMs are given in different levels of abstraction - most in a high level).

We also saw an example of a language that is not decidable:

$$D = \{\langle p \rangle \mid p \text{ is a polynomial with an integral root}\}$$

More Decidable Languages

We give more examples of decidable languages.

These languages represent problems in the domain of automata and grammars.

These problems are related to important applications such as parsing of programming languages

The problems given are related to Regular Languages and to Context Free Languages

Decidable problems concerning regular languages

Consider the following language;

$$A_{DFA} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$$

The problem of testing if a DFA B accepts a string w is the same of testing whether $\langle B, w\rangle$ belongs to the language A_{DFA}.

We can formulate other computational problems in term of testing membership in a language.

Showing that a language is decidable is the same as showing that the related computational problem is decidable.

Theorem: A_{DFA} is decidable.

Proof: Se proof of theorem 4.1 in Chapter 4 of Sipser's book.

Decidable problems concerning regular languages

A similar result but now for nondeterministic finite automata (NFA):

$$A_{NFA} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts input string } w\}$$

Theorem: A_{NFA} is decidable.

Proof: Se proof of theorem 4.2 in Chapter 4 of Sipser's book.

And for regular expressions:

$$A_{REX} = \{R, w\rangle \mid R \text{ is a regular expressions that generates string } w\}$$

Theorem: A_{REX} is decidable.

Proof: See proof of theorem 4.3 of Sipser's book.

Decidable problems concerning regular languages

Determine whether a finite automata accepts any strings at all:

$$E_{DFA} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$$

Theorem: E_{DFA} is decidable.

Proof: Se proof of theorem 4.4 in Chapter 4 of Sipser's book.

The problem of determining whether two DFAs recognize the same language is also decidable:

$$EQ_{DFA} = \{\langle A, B \rangle \mid A, B \text{ are DFAs and } L(A) = L(B)\}$$

Theorem: EQ_{DFA} is decidable.

Proof: See proof of theorem 4.5 of Sipser's book.

Contents

Undecidable problems

The problem of determining whether a TM M accepts a given input string w is undecidable.

Phrasing the sentence above in terms of languages we say that the language A_{TM} below is undecidable.

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and M accepts w}\}$$

Theorem: 4.11 The language A_{ATM} is undecidable.

Proof: We will come back to the proof later....

Undecidable problems

Before proving the Theorem 4.11 that says the language A_{TM} is undecidable observe that language A_{TM} is **Turing-recognizable**.

The following TM U **recognizes** language A_{TM}:

U = *"On input* $\langle M, w \rangle$*, where* M *is a TM and* w *is a string:*

1. *Simulate* M *on input* w
2. *If* M *ever enters* q_{accept} *state, accepts; if* M *ever enters* q_{reject} *state, rejects."*

Note that the TM U described above **loops** on input $\langle M, w \rangle$ if M loops on w, that is why U does not decide A_{TM}.

Observe that U is a Universal Turing machine since it can simulate any other TM given its description.

Diagonalization Method

The proof that A_{TM} is undecidable uses a a technique called **diagonalization** discovered by mathematician Georg Cantor in 1873.

If we have two infinite sets, how to tell if one is larger than the other or if they are of the same size? Cantor used the method for measuring the sizes of infinite sets.

For instance, take the sets of even numbers and the set of all string over alphabet $\{0, 1\}$. Both sets are infinite, but is one larger than the other or are they of the same size?

Cantor observed that two sets have the same size if the elements of one set can be paired with the elements of the other

Diagonalization Method

Definition: Assume a function $f : A \to B$.

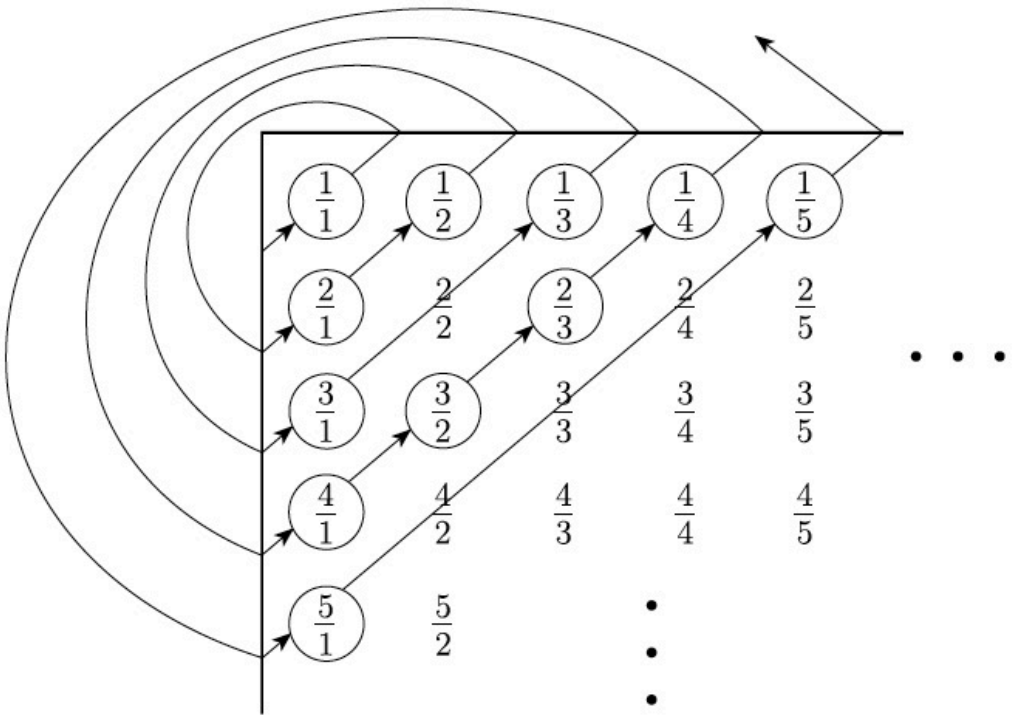
- f is **one-to-one** if it never maps two different elements of to the same place - that is, if $f(a) \neq f(b)$ whenever $a \neq b$;
- f is **onto** if it hits every element of B - that is, if for every $b \in B$ there is an $a \in A$ such that $f(a) = b$.
- We say that A and B are the **same size** if there is a one-to-one and onto function $f : A \to B$.
- A function that is both one-to-one and onto is called a **correspondence**

Example: Let $\mathbb{N}$ the set of natural numbers $\{0, 1, 2, \ldots\}$, and $\mathscr{E}$ the set of even natural numbers $\{2, 4, 6 \ldots\}$. Both sets have the same size. The correspondence f mapping $\mathbb{N}$ to $\mathscr{E}$ is the function $f(n) = 2n$

Definition: A set is **countable** if either it is finite or it has the same size of $\mathbb{N}$.

Diagonalization Method

Let $\mathbb{Q} = \{\frac{m}{n} \mid m, n \in \mathbb{N}\}$ be the set of positive rational numbers. Our intuition says that, since $\mathbb{N} \subset \mathbb{Q}$, the set $\mathbb{Q}$ is much larger than the set $\mathbb{N}$. But $\mathbb{Q}$ is countable! So $\mathbb{Q}$ and $\mathbb{N}$ have the same size!



By disposing the numbers in the matrix above and by following the arrow direction we can place all elements of $\mathbb{Q}$ in a list. After removing repeated numbers we have a **correspondence** mapping $\mathbb{Q}$ to $\mathbb{N}$

Diagonalization Method

A real number is one that has a decimal representation (example: $\pi = 3.1415926\ldots$) With diagonalization Cantor proved that $\mathbb{R}$ is **uncountable**

We assume that there is a correspondence f between $\mathbb{R}$ and $\mathbb{N}$ and we derive a contradiction from this. The table below shows a possible correspondence.

n	$f(n)$
1	3.14159...
2	55.55555...
3	0.12345...
4	0.50000...
⋮	⋮

The proof of the impossibility of any correspondence consists in constructing a real number x and showing that it is **impossible** to place it any line of the table above.

Diagonalization Method

We construct a real number x between 0 and 1 as follows:

- the 1st digit of x after the decimal point should be different from the 1st digit after the decimal point of real number in line 1 of the table
- the 2nd digit of x after the decimal point should be different from the 2nd digit after the decimal point of real number in line 2 of the table
- and so on ....

The number $0,4641\ldots$ below is an example of a real number constructed as above and clearly $\forall n \in \mathbb{N}.\ x \neq f(n)$

n	$f(n)$
1	$3.\underline{1}4159\ldots$
2	$55.5\underline{5}555\ldots$
3	$0.12\underline{3}45\ldots$
4	$0.500\underline{0}0\ldots$
$\vdots$	$\vdots$

$$x = 0.4641\ldots$$

An undecidable language (I)

Back to the proof of Theorem 4.11:

Theorem: 4.11 The language A_{TM} is undecidable.

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$$

Proof: We assume that A_{TM} **is** decidable and we obtain a contradiction.

From the assumption that A_{TM} is decidable, there is a TM, let's call it H, that decides A_{TM}, i.e., TM H always stops, with *accept* if TM M accepts w, or with *reject* if TM M fails to accepts w.

In other words, TM H behaves as follows:

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

An undecidable language (II)

Note that the input for TM H is a string $\langle M, w \rangle$ which is the concatenation of two strings: string $\langle M \rangle$ codifying TM M, and string w which is an input for TM M.

Let's now define another Turing machine D, which, on input $\langle M \rangle$, calls TM H (which we suppose exists) with input $\langle M, \langle M \rangle \rangle$.

That is, D calls H to determine what M does when the input to M is its own description $\langle M \rangle$.

Once H returns this information to D, the TM D returns, as its result, the **opposite** information returned by H.

An undecidable language (III)

D = "On input $\langle M \rangle$*, where* M *is a TM:*
1. *run* H *on input* $\langle M, \langle M \rangle \rangle$
2. *if* H *accepts,* ***reject****; if* H *rejects,* ***accept****"*

We can describe the behaviour of TM D as the following function:

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if TM M does not accept } \langle M \rangle \\ \text{reject} & \text{if TM M accepts } \langle M \rangle \end{cases}$$

But what happens when we run TM D with its own description $\langle D \rangle$ as input? We have the following:

$$D(\langle D \rangle) = \begin{cases} \text{accept} & \text{if TM D does not accept } \langle D \rangle \\ \text{reject} & \text{if TM D accepts } \langle D \rangle \end{cases}$$

And here we have a contradiction! So neither TM D nor TM H exist. Hence A_{TM} is undecidable. □

An undecidable language (IV)

Where was the diagonalization method used in the previous proof?

In the table below we list **all** TMs M_1, M_2, ... in the rows. And we list all their descriptions $\langle M_1 \rangle$, $\langle M_2 \rangle$, ... in the columns.

The entries tell whether the machine in a given row accepts the input in a given column. The entry is **accept** if the TM accepts the input, and the entry is left blank is the TM fails to accept its input (either it rejects or it loops).

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$			$\vdots$		

An undecidable language (IV)

In the following table, the entries are the outputs of TM H when it runs on inputs corresponding to the previous table:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*	
M_4	*accept*	*accept*	*reject*	*reject*	
$\vdots$			$\vdots$		

An undecidable language (V)

Since D is a TM (built based on the supposition that TM H exists) it should be positioned somewhere in the list of TMs M_1, M_2, ... in the rows.

By the definition of D, the entries for it in the table are the opposite of the entries in the table's diagonal.

But then it is impossible to fill in the position in the table marked with the question mark !! (a contradiction)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*		*accept*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*		*reject*	
M_4	*accept*	*accept*	*reject*	*reject*		*accept*	
$\vdots$		$\vdots$			$\ddots$		
D	*reject*	*reject*	*accept*	*accept*		?	
$\vdots$		$\vdots$					$\ddots$