Monash University
Faculty of Information Technology

FIT2014 Theory of Computation

Lecture 20
Decidability

slides by Graham Farr
based in part on previous slides by David Albrecht

Overview

- Decision problems
- Decidable problems and languages
- Deciders
- Closure

Deciders

Reminder:

A decider is a Turing Machine that halts for every input.

A language is <u>decidable</u> if it is Accept(M) for some <u>decider</u> M.

...in which case, its complement is Reject(M).

Examples:

- Regular Languages
- Context Free Languages
- $\{\mathtt{a}^n\mathtt{b}^n\mathtt{a}^n : n \geq 0\}$

Decidable: synonyms

$$
\begin{aligned}
&\text{decidable} \\
&\quad = \quad \text{recursive} \\
&\quad = \quad \text{solvable} \\
&\quad = \quad \text{computable}
\end{aligned}
$$

. . . sometimes, though "computable" has been used with other meanings too.

Decision Problems

INPUT: an integer
QUESTION: Is it even?

INPUT: a string.
QUESTION: Is it a palindrome?

INPUT: an expression in propositional logic
QUESTION: Is it ever True?

INPUT: a graph G, and two vertices s and t
QUESTION: is there a path from s to t in G ?

INPUT: a Python program
QUESTION: is it syntactically correct?

INPUT: a Finite Automaton
QUESTION: Does it define the empty language?

INPUT: two Regular Expressions
QUESTION: Do they define the same language?

INPUT: a Finite Automaton
QUESTION: Does it define an infinite language?

INPUT: a Context Free Grammar
QUESTION: Does it define the empty language?

INPUT: a Context Free Grammar
QUESTION: Does it generate an infinite language?

INPUT: a Context Free Grammar and a string w
QUESTION: Can w be generated by the grammar?

Decision Problems

A **decision problem** is a problem where, for each input, the answer is Yes or No.

A decider **solves** a decision problem if it

- Accepts an input for which the answer is Yes, and
- Rejects any input for which the answer is No.

Decision problem $\longrightarrow$ language

- $\{$ YES-inputs $\}$

Language $\longrightarrow$ decision problem

- INPUT: a string
  (over some alphabet, usually representing some object)
  QUESTION: Is the string in the Language?

Thus, a decider solves a decision problem if and only if it is a decider for its corresponding language.

Encoding of Input

The input and output for a Turing Machine is always a string.

For any object, O, $\langle O \rangle$ will denote encoding of the object as a string.

If we have several objects, $O_1, \ldots, O_n$, we denote their encoding into a single string by $\langle O_1, \ldots, O_n \rangle$.

Testing Emptiness of Regular Languages

Decision Problem:
INPUT: a Finite Automaton
QUESTION: Does it define the empty language?

Language:

$$\text{FA-Empty} \;:=\; \{\langle A\rangle : A \text{ is a FA } \text{ and } \; L(A) = \emptyset\}$$

Theorem.
FA-Empty is decidable.

Testing Emptiness of Regular Languages

Theorem.
FA-Empty is decidable.

Proof. (outline)

Algorithm:

Input: $\langle A \rangle$ where A is a Finite Automaton.

1. Mark the Start State of A.
2. Repeat until no new states get marked:
   - Mark any state that has a transition coming into it from any state that is already marked.
3. If no final state is marked, Accept; otherwise Reject.

□

Testing Equivalence of Regular Expressions

Decision Problem:

REGULAR EXPRESSION EQUIVALENCE
INPUT: two Regular Expressions
QUESTION: Do they define the same language?

For a Regular expression R, let $L(R)$ be the language defined by R.

Language:

$$\text{RegExpEquiv} := \{\langle A, B\rangle : A, B \text{ are regular expressions and } L(A) = L(B)\}$$

Theorem.
RegExpEquiv is decidable.

Testing Equivalence of Regular Expressions

Proof.

Algorithm:

Input: $\langle A, B \rangle$ where A and B are regular expressions

1. Construct a FA, C, that defines the language

$$(L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B)).$$

2. Run the previous Turing Machine, T, on C.
3. If T accepts C, then Accept, else Reject.

$\square$

Testing Emptiness of Context Free Language

Decision Problem:

INPUT: a Context Free Grammar
QUESTION: Does it define the empty language?

Language:

$$\text{CFG-Empty} := \{G : G \text{ is a CFG and } G \text{ defines the empty language}\}$$

Theorem.
CFG-Empty is decidable.

Testing Emptiness of Context Free Language

Algorithm:

Input: $\langle A \rangle$ where A is a Context Free Grammar.

1. Mark all the terminal symbols in A.
2. Repeat until no new symbols get marked:
   - Mark any non-terminal X that has a production which has all the right-hand symbols marked.
3. If Start Symbol is not marked, Accept, else Reject.

Some Decidable Problems

INPUT: a Finite Automaton
QUESTION: Does it define the empty language?

INPUT: two Regular Expressions
QUESTION: Do they define the same language?

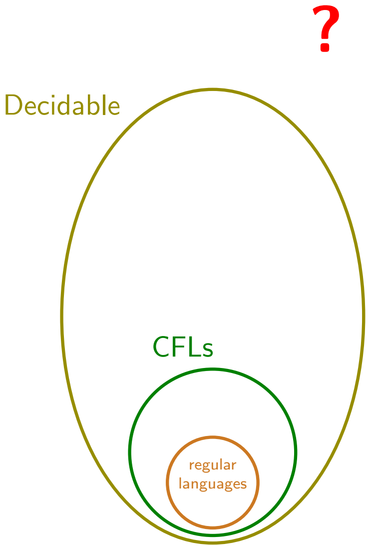
INPUT: a Finite Automaton
QUESTION: Does it define an infinite language?

INPUT: a Context Free Grammar
QUESTION: Does it define the empty language?

INPUT: a Context Free Grammar
QUESTION: Does it generate an infinite language?

INPUT: a Context Free Grammar and a string w
QUESTION: Can w be generated by the grammar?

Language classes



Closure properties

If L is decidable, then $\overline{L}$ is decidable.
If L_1 and L_2 are decidable, then so are

- $L_1 \cup L_2$
- $L_1 \cap L_2$
- $L_1 L_2$
- $\ldots$

Exercise:
Formulate and prove more closure results.

Revision

- Decidable Problems, decidable languages, and the link between them.
- Decision problems, relationship with languages
- Examples of Decidable Problems.
- Closure properties

Reading: Sipser, Section 4.1, pp. 190–201.

Preparation: Sipser, Section 4.2, pp. 201–213, especially pp. 207–209.