Léonard Brice, Mathieu Sassolas
leonard.brice@ulb.be, mathieu.sassolas@ulb.be

Session 3:
Undecidability and reductions

Reminders

A language (or a problem) is *decidable*, or *recursive*, if there is a Turing machine that decides it. We write **R** the set of recursive languages (or problems). A language (or a problem) is *recursively enumerable* if there is a Turing machine that can enumerate it; or equivalently, if there is a Turing machine that recognizes it. We write **RE** the set of recursively enumerable languages (or problems). Note that the class **R** is strictly included in **RE**.

A is *reducible* to B (or *reduces* to B) if there exists a computable mapping f that maps each instance i of A to an instance $f(i)$ of B, such that i is a positive instance of A if and only if $f(i)$ is a positive instance of B. Then, if the problem B is decidable, so is the problem A; or, by contraposition, if the problem A is undecidable, so is the problem B.

1 Decidable problems

1. Prove that the following problem is decidable (an intuitive algorithm is sufficient, as long as it is rigorously explained):

$$\textsc{Accessibility} : \begin{cases} \textit{Input:} & \text{A directed graph } G \text{ (encoded by its adjacency matrix) and two sets of vertices } X \text{ and } Y \\ \textit{Question:} & \text{Does there exist a path in } G \text{ from a vertex of } X \text{ to a vertex of } Y? \end{cases}$$

2. Prove that the following problem:

$$\textsc{RegNonEmptiness} : \begin{cases} \textit{Input:} & \text{A non-deterministic finite automaton } A \\ \textit{Question:} & \text{Do we have } \mathcal{L}(A) \neq \emptyset? \end{cases}$$

   reduces to Accessibility.

3. What can be deduced about the problem RegNonEmptiness?

2 Acceptance and halting problems

In the coming questions, we aim at showing that the acceptance problem of Turing machines:

$$\mathrm{A_{TM}} : \begin{cases} \textit{Input:} & \text{A Turing machine } M \text{ and a word } w \\ \textit{Question:} & \text{Does } M \text{ accept } w? \end{cases}$$

and the halting problem of Turing machines:

$$\textsc{Halt} : \begin{cases} \textit{Input:} & \text{A Turing machine } M \text{ and a word } w \\ \textit{Question:} & \text{Does } M \text{ halt on } w? \end{cases}$$

are undecidable.

1. Assume that $\mathrm{A_{TM}}$ is decidable. Explain why the following problem is decidable:

$$\textsc{Diagonal} : \begin{cases} \textit{Input:} & \text{A Turing machine } M \\ \textit{Question:} & \text{Does } M \text{ reject the encoding of } M? \end{cases}$$

2. Deduce undecidability for A_{TM}.

3. Let M be a Turing machine, and w be a word. Define a Turing machine M' and a word w' such that M accepts w if and only if M' halts on w'.

4. Prove that HALT is undecidable.

3 Reductions

Check whether the following problems are decidable. If they are, describe a Turing machine or an algorithm in pseudo-code that decides them. If not, prove it via reduction from an undecidable problem.

1. $$\text{RIEMANN} : \begin{cases} \textit{Input:} & \text{A word } w \\ \textit{Question:} & \text{Is Riemann's hypothesis}^1 \text{ true?} \end{cases}$$

2. $$\text{INCLUSION}_{\text{TM}} : \begin{cases} \textit{Input:} & \text{Two Turing machines } M_1 \text{ and } M_2 \\ \textit{Question:} & \text{Does } \mathcal{L}(M_1) \subseteq \mathcal{L}(M_2) \text{ hold?} \end{cases}$$

3. $$\text{INCLUSION}_{\text{aut}} : \begin{cases} \textit{Input:} & \text{Two deterministic finite automata } A_1 \text{ and } A_2 \\ \textit{Question:} & \text{Does } \mathcal{L}(A_1) \subseteq \mathcal{L}(A_2) \text{ hold?} \end{cases}$$

4. $$\text{EMPTYWORD} : \begin{cases} \textit{Input:} & \text{A Turing machine } M \\ \textit{Question:} & \text{Question: Does } \mathcal{L}(M) \text{ contain the empty word?} \end{cases}$$

5. $$\text{BOUNDEDSPACE} : \begin{cases} \textit{Input:} & \text{A Turing machine } M \text{ and a word } w \\ \textit{Question:} & \text{Is there an integer } B \text{ such that running } M \text{ on input } w \text{ does not use more than } B \text{ cells of the tape?} \end{cases}$$

6. $$\text{PYTHONZERODIV} : \begin{cases} \textit{Input:} & \text{A Python function } \texttt{f} \\ \textit{Question:} & \text{Are there arguments such that } \texttt{f} \text{ throw a } \texttt{ZeroDivisionError} \text{ exception?} \end{cases}$$

   (We assume here that the code can be ran on a computer with arbitrary memory; in particular, there is no limit on the size of the arguments.)

[1] Riemann's hypothesis states that *Riemann's zeta function*:

$$\zeta : \begin{cases} \mathbb{C} & \to \mathbb{C} \\ s & \mapsto \sum_{n \geq 1} \frac{1}{n^s} \end{cases}$$

is such that every non-trivial zero s of ζ has real part $\Re(s) = \frac{1}{2}$. The study of the zeta function is strongly linked to the study of prime numbers and, therefore, to cryptography. Proposed by Bernhard Riemann in 1859, that conjecture is still open today. The Clay Mathematics Institute offers US$1 million to anyone who would solve it.