Theory of Computation
Homework 2

Tom Nonnenmacher (325341), Sebastian Maier (327504),
Sébastien Delsad (326423), Jérémy Chaverot (315858).

April 2022

To be rigorous, one should verify that the inputs are indeed the descriptions of DFAs or TMs, using the definitions (respectively 5-tuple & 7-tuple), because input strings can be any string in $\{0, 1\}^\star$. This verification can always be done on finite input strings. In this homework, we assume the input is in the correct form (either DFA or TM).

Exercice 1

Let L_1 denote the following language :

$$L_1 = \{ \langle D, D' \rangle : D \text{ and } D' \text{ are DFAs and } L(D) \subseteq L(D') \}.$$

Let us prove that the language L_1 is Turing-decidable, and thus is Turing-recognizable. For this purpose, we use the following equivalences :

$$L(D) \subseteq L(D') \Leftrightarrow L(D) \cap L(D') = L(D) \Leftrightarrow \{L(D) \cap L(D')\} \oplus L(D) = \emptyset$$

We give the description of a Turing machine TM_{SUBSET} :

- get on input a pair $\langle D, D' \rangle$ two DFAs encoded in binary (if the input is not such a tuple, *reject*)
- build a DFA $\mathcal{D}_1$ that accepts the language $L(D) \cap L(D')$ since regular languages are closed under intersection.
- simulate TM_{EQ} with $\langle D, \mathcal{D}_1 \rangle$
- if TM_{EQ} accepts, then *accept*, otherwise if TM_{EQ} rejects, then *reject*.

The Turing machine TM_{EQ}, that verifies whether two DFAs accept exactly the same language or not, is constructed as follows :

- get on input a pair $\langle D, D' \rangle$ two DFAs encoded in binary (if the input is not such a tuple, *reject*)

- build a DFA $\mathcal{D}_2$ that accepts the language $L(D) \oplus L(D')$, that is equivalent to $\{L(D) \cap \overline{L(D')}\} \cup \{\overline{L(D)} \cap L(D')\}$, since regular languages are closed under complement, intersection and union.
- simulate TM_{EMPTY} with $\langle \mathcal{D}_2 \rangle$
- if TM_{EMPTY} accepts, then *accept*, otherwise if TM_{EMPTY} rejects, then *reject*.

The Turing machine TM_{EMPTY}, that verifies whether the language of a DFA is empty or not, is constructed as follows :

- get on input a singleton $\langle D \rangle$ a DFA encoded in binary (if the input is not such a tuple, *reject*)
- given $\langle D \rangle$ with $D = (Q, \Sigma, \delta, q_0, F)$
  1. Initialize $R = \{q_0\}$
  2. For each $q \in R$ and $q' \in Q \setminus R$, check if there exists a transition of the form $\delta(q, a) = q'$ for some $a \in \Sigma$.
  3. If at least one such q' is found, add q' to R and go back to step 2.
  4. *Accept* iff $R \cap F = \emptyset$.

All of this is summarized in the figures below.

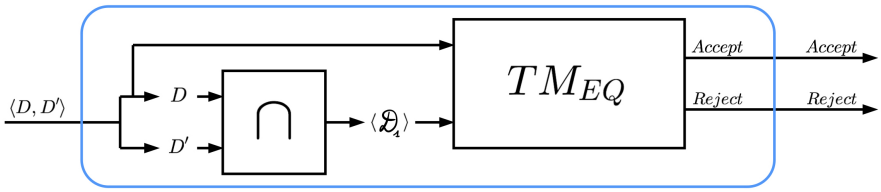

Figure 1: Caption of Turing machine TM_{SUBSET}

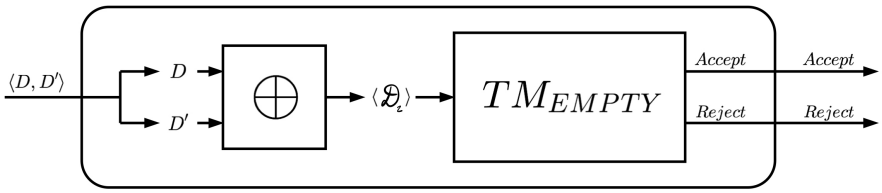

Figure 2: Caption of Turing machine TM_{EQ}

This completes our construction, and TM_{SUBSET} decides L_1 (and also recognizes L_1 as a consequence).

Let L_2 denote the following language :

$$L_2 \;=\; \{\, \langle M \rangle \;:\; M \text{ is a TM and } L(M) \text{ is infinite} \,\}.$$

Let us show the unrecognizability of L_2. First of all, we give the definition of the language A_{TM} :

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

It was shown in class that this language is Turing-undecidable (proof by contradiction using diagonalization), but it is easy to convince ourselves it is Turing-recognizable, using the following TM :

> 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 accepts, *accept*.
> 3. If M does not accept w, then *reject* or *loop*.
> 4. This is why this TM is only a recognizer, not a decider.

This implies that $\overline{A_{TM}}$ is not Turing-recognizable. Now, it is enough to show that language $\overline{A_{TM}}$ mapping reduces to language L_2, written $\overline{A_{TM}} \leq_m L_2$. Below we present a suitable reduction.

- We define a function $f(\langle T,\, w \rangle) = \langle M \rangle$, where M is a TM which performs the following :
  1. On input x, simulate T with w for |x| steps.
  2. If T accepts w, go into an *infinite loop*.
  3. Otherwise *accept* x.

Thus, we have that :

- If T accepts w (namely $\langle T,\, w \rangle \notin \overline{A_{TM}}$), the language of M is empty (it accepts no string) and in particular $\langle M \rangle \notin L_2$.
- If T does not accept w (namely $\langle T,\, w \rangle \in \overline{A_{TM}}$), the language of M is infinite (it accepts all strings) and in particular $\langle M \rangle \in L_2$.

Consequently we obtain the equivalence $\langle T,\, w \rangle \in \overline{A_{TM}} \;\Leftrightarrow\; f(\langle T,\, w \rangle) \in L_2$, and f is a *computable* function.

Therefore, it is a valid reduction, i.e. $\overline{A_{TM}} \leq_m L_2$, and we can infer L_2 is Turing-unrecognizable (and is also Turing-undecidable as a consequence).

Exercice 2

Let A and B be two disjoint languages (i.e. $A \cap B = \emptyset$). Say that a language C *separates* A and B iff $A \subseteq C$ and $B \subseteq \overline{C}$.

2a Suppose that A and B are co-Turing-recognizable (that is, $\overline{A}$ and $\overline{B}$ are Turing-recognizable). Let us show that there is a Turing-decidable language C that separates A and B.

To this end, we define two TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ recognizing $\overline{A}$ and $\overline{B}$ respectively. Let $\mathcal{T}$ be a Turing machine which performs the following on input x :

1. Simulate $\mathcal{M}_1$ and $\mathcal{M}_2$ on x in *parallel*, that means by alternating the steps of the two TMs.
2. If $\mathcal{M}_1$ accepts first, *reject*. If $\mathcal{M}_2$ accepts first, *accept*.

By assumption, we have that $A \cap B = \emptyset$, which is equivalent to $\overline{A} \cup \overline{B} = \Sigma^\star$, and this forces the TM $\mathcal{T}$ to halt for all input x. So at the end, either $\mathcal{M}_1$ or $\mathcal{M}_2$ will accept the input x. We denote C the language decided by the TM $\mathcal{T}$, and we have that :

- If $x \in A$, x will not be recognized by $\mathcal{M}_1$ and will be accepted by $\mathcal{M}_2$ first. Hence $A \subseteq C$.
- If ~~$x \notin A \Rightarrow$~~ $x \in B$, x will not be recognized by $\mathcal{M}_2$ and will be accepted by $\mathcal{M}_1$ first. Hence $B \subseteq \overline{C}$.

Therefore, this completes our construction of a decidable language C that separates A and B.

2b Define two disjoint languages by :

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

$$B = \{\, \langle M, w \rangle \,:\, M \text{ is a TM and } M \text{ rejects } w \,\}.$$

We prove that there does not exist any Turing-decidable language C that separates A and B, that is for all language C separating A and B, C is always non-Turing-decidable.

It appears that A and B are Turing-undecidable but Turing-recognizable, since for both of them it is possible to build a TM recognizing them (c.f. the language A_{TM} in exercice 1).

Note that given the definitions of A and B, the two languages are not *exhaustive* because there exist TMs that neither accept nor reject w (a looping behavior for exemple), hence $A \cup B \neq \Sigma^\star$.

We consider $U = \overline{A \cup B} = \overline{A} \cap \overline{B}$. As a Turing machine can only accept, reject or loop, the language U is such that :

$$U = \{\, \langle M, w \rangle \,:\, M \text{ is a TM and } M \text{ loops on } w \,\}$$

Indeed, it does not contain the TMs that accept w (actually A), and it does not contain the TMs that reject w (actually B) so it can only contain language that loop on w.

Now, consider C that separates A and B, i.e. $A \subseteq C$ and $B \subseteq \overline{C} \Rightarrow C \subseteq \overline{B}$. Thus we have that $C = A \cup V$ where V is a language such that $V \subseteq U$.

First case Let $V = \emptyset$. Hence $C = A \cup \emptyset = A$. As we know, A is in fact exactly A_{TM} which is Turing-undecidable. So C is Turing-undecidable.

Second case Let $V \neq \emptyset$. Then C is a language such that :

$$\begin{aligned} C = &\{ \langle M, w \rangle \, : \, M \text{ is a TM and } M \text{ accepts } w \} \\ &\cup \{ \langle M, w \rangle \, : \, M \text{ is a TM and } M \text{ loops on } w \} \end{aligned}$$

Which is similar to :

$$C = \{ \langle M, w \rangle \, : \, M \text{ is a TM and } M \text{ accepts } w \text{ or } M \text{ loops on } w \}$$

As $V \neq \emptyset$, there exist TMs in C which loop on a string, and as we cannot decide if a TM can loop on a string, that means C is undecidable. Quod Erat Demonstrandum.

Here is the scheme of another proof. Suppose towards contradiction that there exists a decidable language C that separates A and B. We denote $\mathcal{D}$ the TM which is a decider for C, and performs as follows on input string x:

1. Get its own binary description $\langle \mathcal{D} \rangle$.
2. Simulate $\mathcal{D}$ on w.
3. Verify if $\langle \mathcal{D}, w \rangle$ belongs to C. If it's the case, *reject*. If not, *accept*.

If the decider $\mathcal{D}$ accepts the input w, it means $\langle \mathcal{D}, w \rangle \in A \subseteq C$, however $\mathcal{D}$ rejects w then. If the decider $\mathcal{D}$ rejects the input w, it means $\langle \mathcal{D}, w \rangle \in B \subseteq \overline{C}$, however $\mathcal{D}$ accepts w then. It is obviously a contradiction and the language C cannot be decidable.

[...]