CS420

Introduction to the Theory of Computation

Lecture 20: Undecidable problems

Tiago Cogumbreiro

Today we will learn...

Decidability of

- The Halting Problem
- Emptiness for TM
- Regularity
- Equality

> Section 5.1

Recap Lesson 16/17

Decidable languages:

- $A_{DFA}, A_{REX}, A_{NFA}, A_{CFG}$

```
def A_DFA(D, w):
  return D accepts w
```

$$A_{DFA} = \{\langle D, w\rangle \mid D \text{ accepts } w\}$$

- E_{DFA}, E_{CFG}

```
def E_DFA(D):
  return L(D) = {}
```

$$E_{DFA} = \{\langle D\rangle \mid L(D) = \emptyset\}$$

- EQ_{DFA}

```
def EQ_DFA(D1, D2):
  return L(D1) = L(D2)
```

$$EQ_{DFA} = \{\langle N_1, N_2\rangle \mid L(N_1) = L(N_2)\}$$

Exercise 1

Prove or falsify the following statement: EQ_{REX} is undecidable.

Exercise 1

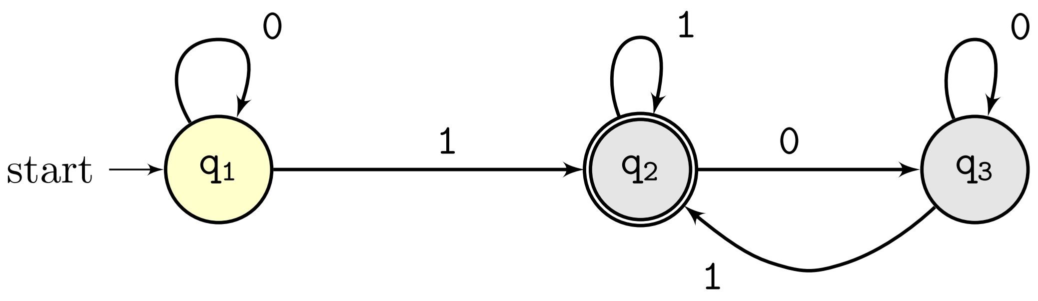
Prove or falsify the following statement: EQ_{REX} is undecidable.

Proof. False. EQ_{REX} is decidable, as given by the following pseudo code, where `EQ_DFA` is the decider of EQ_{DFA} and `REX_TO_DFA` is the conversion from a regular expression into a DFA.

```
def EQ_REX(R1, R2):
  return EQ_DFA(REX_TO_DFA(R1), REX_TO_DFA(R2))
```

Exercise 2

Let D be the DFA below




```
def A_DFA(D, w): return D accept w
def E_DFA(D): return L(D) == {}
def EQ_DFA(D1, D2): return L(D1) == L(D2)
```

- Exercise 2.1: Is $\langle D, 0100\rangle \in A_{DFA}$?
- Exercise 2.2: Is $\langle D, 101\rangle \in A_{DFA}$?
- Exercise 2.3: Is $\langle D\rangle \in A_{DFA}$?
- Exercise 2.4: Is $\langle D, 101\rangle \in A_{REX}$?
- Exercise 2.5: Is $\langle D\rangle \in E_{DFA}$?
- Exercise 2.6: Is $\langle D, D\rangle \in EQ_{DFA}$?
- Exercise 2.7: Is $101 \in A_{REX}$?

Exercise 3

Recall that DFAs are closed under $\cap$. Prove the following statement.

If A is regular, then X_A decidable.

$$X_A = \{\langle D \rangle \mid D \text{ is a DFA} \land L(D) \cap A \neq \emptyset\}$$

Exercise 3

Recall that DFAs are closed under $\cap$. Prove the following statement.

If A is regular, then X_A decidable.

$$X_A = \{\langle D \rangle \mid D \text{ is a DFA} \wedge L(D) \cap A \neq \emptyset\}$$

Proof. If A is regular, then let C be the DFA that recognizes A. Let `intersect` be the implementation of $\cap$ and `E_DFA` the decider of E_{DFA}. The following is the decider of X_A.

```
def X_A(D):
  return not E_DFA(intersect(C, D))
```

Theorem 4.22

L decidable iff L recognizable and L co-recognizable

Theorem 4.22

L decidable iff L recognizable and L co-recognizable

Proof. We can divide the above theorem in the following three results.

1. If L decidable, then L is recognizable. **(Proved.)**
2. If L decidable, then L is co-recognizable. **(Proved.)**
3. If L recognizable and L co-recognizable, then L decidable.

Part 3. If L recognizable and $\overline{L}$ recognizable, then L decidable.

We need to extend our mini-language of TMs

`plet b ← P1 \\ P2 in P3`

Runs P1 and P2 in parallel.

- If P1 and P2 loop, the whole computation loops
- If P1 halts and P2 halts, pass the success of both to P3
- If P1 halts and P2 loops, pass the success of P1 to P3
- If P1 loops and P2 halts, pass the success of P2 to p3

```
Inductive par_result :=
| pleft: bool → par_result
| pright: bool → par_result
| pboth: bool → bool → par_result.
```

Part 3. If L recognizable and $\overline{L}$ recognizable, then L decidable.

Proof.

1. Let M_1 recognize L from assumption L recognizable
2. Let M_2 recognize $\overline{L}$ from assumption $\overline{L}$ recognizable
3. Build the following machine

```
Definition par_run M1 M2 w :=                (* M1 and M2 are parameters of the machine *)
  plet b ← Call M1 w \\ Call M2 w in         (* Call M1 with w and M2 with w in parallel *)
  match b with
  | pleft true    ⇒ ACCEPT
  | pboth true _  ⇒ ACCEPT                   (* If M1 accepts w, accept *)
  | _ ⇒ REJECT                               (* Otherwise, reject *)
  end.
```

4. Show that `par_run M1 M2` recognizes L and is a decider.

Part 3. If L recognizable and $\overline{L}$ recognizable, then L decidable.

Point 4: Show that `par_run M1 M2` recognizes L and is a decider.

- 1. Show that `par_run M1 M2` recognizes L: `par_run M1 M2` accepts w iff $L(w)$
- 1.1. `par_run M1 M2` accepts w, then $w \in L$
- 1.2. $w \in L$, then `par_run M1 M2` accepts w case analysis on run `M2` with w

```
Definition par_run M1 M2 w :=
  plet b ← Call M1 w \\ Call M2 w in
  match b with
  | pleft true
  | pboth true _ ⇒ ACCEPT
  | _ ⇒ REJECT
  end.
```

- `M1` recognizes L
- `M2` recognizes $\overline{L}$
- Lemma `par_mach_lang`

Part 3. If L recognizable and $\overline{L}$ recognizable, then L decidable.

Point 4: Show that `par_run M1 M2` recognizes L and is a decider.

1. Show that `par_run M1 M2` recognizes L: `par_run M1 M2` accepts w iff $L(w)$
   1. `par_run M1 M2` accepts w, then $w \in L$ by case analysis on `Call M1 w \\ Call M2 w`:
      - `pleft true` and `M1` accepts w: holds since `M1` recognizes L
      - `pboth true _` and `M1` accepts w: same as above
      - otherwise: contradiction
   2. $w \in L$, then `par_run M1 M2` accepts w case analysis on run `M2` with w
      - `M2` accept w: `par_run M1 M2` accept since `M1`accepts with w
      - `M2` loops w: `par_run M1 M2` accept since `M1` accepts with w
      - `M2` reject w: `par_run M1 M2` accept since `M1` accepts with w

Part 3. If L recognizable and $\overline{L}$ recognizable, then L decidable.

Point 4: Show that `par_run M1 M2` recognizes L and is a decider.

2. Show that `par_run M1 M2` decides L

   (Walk through the proof of `recognizable_co_recognizable_to_decidable`...)