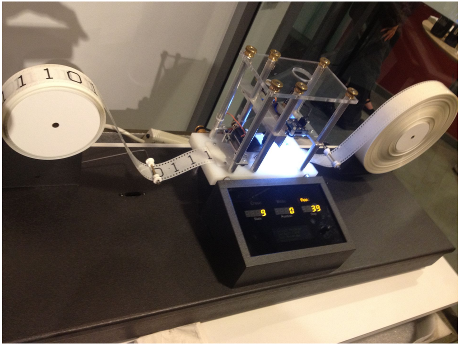
COS210 - Theoretical Computer Science
Decidable and Undecidable Languages: Part 3

Difference between M and $\langle M \rangle$

String $\langle M \rangle$ that describes M

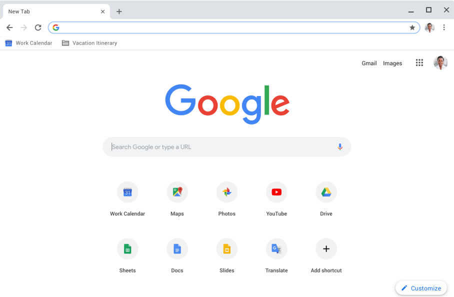
Turing machine M



$$q_0 0\$ \to q_0 R\$SS$$
$$q_0 0S \to q_0 RSSS$$
$$q_0 1\$ \to q_0 N\$$$
$$q_0 1S \to q_1 R\epsilon$$
$$q_0 \Box\$ \to q_0 N\epsilon$$
$$q_0 \Box S \to q_0 NS$$
$$q_1 0\$ \to q_1 N\$$$
$$q_1 0S \to q_1 NS$$
$$q_1 1\$ \to q_1 N\$$$
$$q_1 1S \to q_1 R\epsilon$$
$$q_1 \Box\$ \to q_1 N\epsilon$$
$$q_1 \Box S \to q_1 NS$$

Difference between P and $\langle P \rangle$

Program P



Source code $\langle P \rangle$ of the program

```
QModelIndex start;
if (currentIndex().isValid())
    start = currentIndex();
else
    start = d->model->index(0, 0, d->root);

bool skipRow = false;
bool keyboardTimeWasValid = d->keyboardInputTime.isValid();
qint64 keyboardInputTimeElapsed = d->keyboardInputTime.restart();
if (search.isEmpty() || !keyboardTimeWasValid
    || keyboardInputTimeElapsed > QApplication::keyboardInputInterval()) {
    d->keyboardInput = search;
    skipRow = currentIndex().isValid(); //if it is not valid we should real
} else {
    d->keyboardInput += search;
}
```

Set of Turning Machines

We have seen two examples of **undecidable languages:**

$$A_{TM} = \{\langle M, w\rangle : M \text{ is a } \textbf{Turing machine} \text{ that } \textbf{accepts } w\}$$
$$Halt = \{\langle P, w\rangle : P \text{ is a } \textbf{program} \text{ that } \textbf{terminates} \text{ on input } w\}$$

We want to show that there are many languages involving Turing machines that are undecidable

For this will define $\mathcal{T}$ to be the language of all binary encodings of all Turning machines, or:

$$\mathcal{T} = \{\langle M\rangle : M \text{ is a Turing machine with input alphabet } \{0, 1\}\}$$

$\mathcal{T}$ is actually **decidable** under any reasonable encoding of a Turing machine:

It is possible to determine within a limited amount of time whether a sting is a valid encoding of a Turing machine or not

Set of Turning Machines

The question that is slightly harder is, given a subset $\mathcal{P} \subset T$:

- Is the language $\mathcal{P}$ **decidable**?
- In particular, can we build a **decision procedure** that will tell us if a given Turing machine M is in $\mathcal{P}$ or not
- This is where **Rice's Theorem** becomes useful

Rice's Theorem

Theorem (Rice's Theorem)

Let $\mathcal{P}$ be a subset of $\mathcal{T}$ such that:

1. $\mathcal{P} \neq \emptyset$ *(at least one $\langle M \rangle$ is contained in $\mathcal{P}$)*
2. $\mathcal{P}$ *is a* **proper subset** *of $\mathcal{T}$ ($\mathcal{P} \neq \mathcal{T}$)*
3. *For any two Turing machines M_1 and M_2 with $L(M_1) = L(M_2)$*
   - **either both** *encodings $\langle M_1 \rangle$ and $\langle M_2 \rangle$ are in $\mathcal{P}$*
   - **or none** *of the encoding $\langle M_1 \rangle$ and $\langle M_2 \rangle$ is in $\mathcal{P}$*

Then the language $\mathcal{P}$ is **undecidable**

Rice's Theorem

$\mathcal{P}$ can be seen as the set of machines that **satisfy a certain property**:

- Conditions 1 says that **at least one** Turing machine satisfies the property
- Condition 2 says that **not all** Turning machines satisfy the property
- Condition 3 says that for any machine M, whether M satisfies the property or not **depends on the language** $L(M)$

We can distinguish between two different **types of properties**:

- **Semantic properties** (undecidable) are about the machine's behaviour. For instance, does the machine machine accept the input 1011?
- **Syntactic properties** (decidable) are about the structure of a machine. For instance, does the machine have five state?

Rice's Theorem: Examples

Consider the following languages:

- $\mathcal{P}_1 = \{\langle M \rangle : M \text{ is a Turing machine and } \epsilon \in L(M)\}$
- $\mathcal{P}_2 = \{\langle M \rangle : M \text{ is a Turing machine and } L(M) = \{1011, 001100\}\}$
- $\mathcal{P}_3 = \{\langle M \rangle : M \text{ is a Turing machine and } L(M) \text{ is regular}\}$
- $\mathcal{P}_4 = \{\langle M \rangle : M \text{ is a Turing machine and } L(M) = \emptyset\}$

We will show that the language $\mathcal{P}_3$ satisfies the conditions of Rice's Theorem

Can you show that other languages satisfy the conditions of Rice's Theorem as well?

Rice's Theorem:Examples

$$\mathcal{P}_3 = \{\langle M \rangle : M \text{ is a Turing machine and } L(M) \text{ is a } \textbf{regular language}\}$$

First, we show that $\mathcal{P}_3$ is not empty by providing an element of $\mathcal{P}_3$

- We know how to construct a DFA that accepts the language $L = \{10\}$
- Hence, the L **regular**
- We also know how to construct a Turing machine M with $L(M) = L$
- Therefore $\langle M \rangle \in \mathcal{P}_3$ and $\mathcal{P}_3 \neq \emptyset$

Second, we show that $\mathcal{P}_3 \neq \mathcal{T}$ by providing an element that is not in $\mathcal{P}_3$

- We know that the language $L' = \{a^n b^n : n \geq 1\}$ is **not regular**
- We know how to construct a Turing machine M' with $L(M') = L'$
- Therefore $\langle M' \rangle \notin \mathcal{P}_3$ and $\mathcal{P}_3 \neq \mathcal{T}$

Rice's Theorem:Examples

$$\mathcal{P}_3 = \{\langle M \rangle : M \text{ is a Turing machine and } L(M) \text{ is a } \textbf{regular language}\}$$

Finally, we show that if $L(M_1) = L(M_2)$ then either both $\langle M_1 \rangle$ and $\langle M_2 \rangle$ are in $\mathcal{P}_3$, or both are not in $\mathcal{P}_3$

- We know that if $L(M_1) = L(M_2)$ then $L(M_1)$ is regular if and only if $L(M_2)$ is regular
- Hence, either both languages are regular and both $\langle M_1 \rangle$ and $\langle M_2 \rangle$ are in $\mathcal{P}_3$
- Or both languages are not regular and both $\langle M_1 \rangle$ and $\langle M_2 \rangle$ are not in $\mathcal{P}_3$

Rice's Theorem: Proof Idea

Theorem (Rice's Theorem)

Let $\mathcal{P}$ be a subset of $\mathcal{T}$ such that:

1. $\mathcal{P} \neq \emptyset$ *(at least one $\langle M \rangle$ is contained in $\mathcal{P}$)*
2. $\mathcal{P}$ *is a* **proper subset** *of $\mathcal{T}$ ($\mathcal{P} \neq \mathcal{T}$)*
3. *For any two Turing machines M_1 and M_2 with $L(M_1) = L(M_2)$*
   - **either both** *encodings $\langle M_1 \rangle$ and $\langle M_2 \rangle$ are in $\mathcal{P}$*
   - **or none** *of the encoding $\langle M_1 \rangle$ and $\langle M_2 \rangle$ is in $\mathcal{P}$*

Then the language $\mathcal{P}$ is **undecidable**

Rice's Theorem: Proof Idea

Reduction to the Halting problem:

- Assume that a language $\mathcal{P}$ that satisfies the Conditions 1, 2 and 3 of Rice's theorem is decidable
- Then there exists a Turing machine M that decides $\mathcal{P}$
- It is then possible to construct another machine M' that makes use of M and decides the Halting problem
- But we know that the Halting problem is undecidable
- Contradiction
- A language $\mathcal{P}$ that satisfies the Conditions 1, 2 and 3 of Rice's theorem is undecidable

Enumerability

A language A is **enumerable**, if there exists an algorithm with the following property:

- If $w \in A$, then the algorithm **terminates and accepts** on input w
- if $w \notin A$, then either
  - the algorithm **terminates and rejects** on input w
  - or it **does not terminate** on input w and does not tell us that $w \notin A$

Theorem

Every decidable language is enumerable

Not all undecidable languages are enumerable, but some are

Hilbert's Problem

Hilbert's problem asks whether the language:

$$\textit{Hilbert} = \{\langle p \rangle : \text{p is a polynomial equation that has an integer solution}\}$$

is decidable or not

Example for p:

$$12x^3y^7z^5 + 7x^2y^4z - x^4 + y^2z^7 - z^3 + 10 = 0$$

- In 1970 it was proven that the language *Hilbert* is **undecidable**
- We can prove that *Hilbert* is **enumerable** by defining an algorithm that enumerates equations with integer solutions

Hilbert's problem

The *HILBERT* algorithm is given by:

Algorithm $HILBERT(\langle p \rangle)$:

$n :=$ the number of variables in p;
for each $(x_1, x_2, ..., x_n) \in \mathbb{Z}^n$
  do $R :=$ p$(x_1, x_2, ..., x_n)$;
    if $R = 0$
    then terminate and accept

Example for p: $12x^3y^7z^5 + 7x^2y^4z - x^4 + y^2z^7 - z^3 + 10 = 0$

HILBERT will iterate over solution candidates for x, y, z:
$(0, 0, 0), (0, 0, 1), (0, 0, -1), \ldots$

Hilbert's problem

The *HILBERT* algorithm is given by:

Algorithm $HILBERT(\langle p \rangle)$:

$n :=$ the number of variables in p;
for each $(x_1, x_2, ..., x_n) \in \mathbb{Z}^n$
  do $R :=$ p$(x_1, x_2, ..., x_n)$;
    if $R = 0$
    then terminate and accept

If the equation p has an **integer solution**, then *HILBERT* will eventually find it and **terminate**

This is true because the set of solution candidates $\mathbb{Z}^n$ is **countable**