# Introduction to Theoretical Computer Science

Lecture 7: Undecidability

Dr. Liam O'Connor

University of Edinburgh Semester 1, 2023/2024

# **Computable Functions**

## Definitions

A (total) function  $\mathbb{N} \to \mathbb{N}$  is *computable*<sup>1</sup> if there is an RM/TM which computes *f*, i.e., given an *x* in *R*<sub>0</sub>, leaves *f*(*x*) in *R*<sub>0</sub>.

A *decision problem* is a set *D* and a query subset  $Q \subseteq D$ . A problem is *decidable* or *computable* if  $d \in Q$  is characterised by a computable function  $f : D \to \{0, 1\}$ .

<sup>&</sup>lt;sup>1</sup>sometimes confusingly called *recursive*, but this is old terminology.

# **Computable Functions**

## Definitions

A (total) function  $\mathbb{N} \to \mathbb{N}$  is *computable*<sup>1</sup> if there is an RM/TM which computes f, i.e., given an x in  $R_0$ , leaves f(x) in  $R_0$ .

A *decision problem* is a set *D* and a query subset  $Q \subseteq D$ . A problem is *decidable* or *computable* if  $d \in Q$  is characterised by a computable function  $f : D \to \{0, 1\}$ .

Note that our *language* problems, for DFAs and CFGs etc., are decision problems where  $D = \Sigma^*$  and Q is the language in question. Also, consider  $D = \mathbb{N}$  and Q = Primes. Or D = RMs and Q = the halting RMs.

<sup>&</sup>lt;sup>1</sup>sometimes confusingly called *recursive*, but this is old terminology.

## **Closure Properties**

#### Are the decidable languages closed under:

- Union?
- Intersection?
- Complement?

## **Closure Properties**

#### Are the decidable languages closed under:

- Union?
- Intersection?
- Complement?
- (yes)

Oracles

Rice's Theorem

## Undecidability

We know that undecidable problems exist, like H.

#### Another Example

 $A_{RM} = \{ \langle \ulcorner M \urcorner, w \rangle \mid M \text{ accepts } w \}$ The proof, in Sipser for TMs, is analogous to our proof for *H*.

Oracles

Rice's Theorem

# Undecidability

We know that undecidable problems exist, like *H*.

## Another Example

 $A_{RM} = \{ \langle \ulcorner M \urcorner, w \rangle \mid M \text{ accepts } w \}$ The proof, in Sipser for TMs, is analogous to our proof for *H*.

#### Aside

We can also use a counting argument. The set of RMs is enumerable, but the set of languages is uncountable. So there are languages that are not decided (or even recognised) by any RM.

How would we show that other problems are undecidable?

# Reductions

A *reduction* is a transformation from one problem to another.

To prove that a problem  $P_2$  is *hard*, show that there is an *easy* reduction from a known hard problem  $P_1$  to  $P_2$ .

#### Therefore

To prove that a problem  $P_2$  is undecidable, show that there is a computable reduction from a known undecidable  $P_1$  to  $P_2$ .

## Pay close attention to the direction of the proof!

Oracles

Rice's Theorem

## A correct example

Suppose it is well known that Hyunwoo cannot lift a car.

Theorem

Hyunwoo cannot lift a loaded truck.

Oracles

Rice's Theorem

## A correct example

Suppose it is well known that Hyunwoo cannot lift a car.

#### Theorem

Hyunwoo cannot lift a loaded truck.

#### Proof

By reduction from the car-lifting problem  $(P_1)$ . Suppose Hyunwoo could lift a loaded truck. Then, he could lift a car by putting the car in the truck and then lifting the truck. But, it is known that Hyunwoo cannot lift a car.

#### Known Hard Problem $\longrightarrow$ New Problem

Oracles

Rice's Theorem

## An **incorrect** example

#### Suppose it is well known that Hyunwoo cannot lift a car.

#### Theorem

Hyunwoo cannot lift a feather.

## An **incorrect** example

#### Suppose it is well known that Hyunwoo cannot lift a car.

#### Theorem

Hyunwoo cannot lift a feather.

#### Proof

By reduction to the car-lifting problem. We can reduce the feather-lifting problem to the car-lifting problem by putting the feather in the car.

It is known that Hyunwoo cannot lift a car. Therefore, Hyunwoo cannot lift a feather (???!).

## Reductions

A Turing Transducer is a RM (or TM) which takes an instance d of a problem  $P_1 = (D_1, Q_1)$  in  $R_0$  and halts with an instance d' = f(d) of  $P_2 = (D_2, Q_2)$  in  $R_0$ . Thus, f is a computable function  $D_1 \rightarrow D_2$ .

A Turing Transducer is a RM (or TM) which takes an instance d of a problem  $P_1 = (D_1, Q_1)$  in  $R_0$  and halts with an instance d' = f(d) of  $P_2 = (D_2, Q_2)$  in  $R_0$ . Thus, f is a computable function  $D_1 \rightarrow D_2$ .

## Definition

A *mapping reduction* (or *many-one* reduction) from  $P_1$  to  $P_2$  is a Turing transducer f as above such that  $d \in Q_1$  iff  $f(d) \in Q_2$ 

If A is mapping reducible to B, and A is undecidable, then B is undecidable.

Example

Oracles 000

# $\mathsf{NotEmpty}_{\mathsf{TM}} = \{ \ulcorner M \urcorner \mid \mathcal{L}(M) \neq \emptyset \}$

#### Example (Proof)

We sketch a mapping reduction from  $A_{TM}$  to NotEmpty<sub>TM</sub>. Given an instance  $\langle M, w \rangle$  of  $A_{TM}$ , our reduction constructs a machine M' whose language is either  $\{w\}$  or  $\emptyset$ . Given input x, it will reject if  $x \neq w$ , else run M on w.

Note that  $\langle M, w \rangle \in A_{\mathsf{TM}}$  iff  $M' \in \mathsf{NotEmpty}_{\mathsf{TM}}$ .

Thus, if we could solve  $NotEmpty_{TM}$  we could solve  $A_{TM}$ , which we know is undecidable. Thus  $NotEmpty_{TM}$  too is undecidable.

Oracles

## **Uniform Halting**

 $UH = \{ \ulcorner M \urcorner | M \text{ halts on all inputs} \}$ 

#### Example (Proof)

We reduce from H to UH. Given a machine M and input w, build a machine M' which ignores its input, writes w to the tape, and then behaves as M. Then M' halts on any input iff M halts on w.

Oracles ●00 Rice's Theorem

# The Looping Problem

## Let L be the subset of RMs (or TMs) that go into an infinite loop. Show that L is undecidable.

## The Looping Problem

Let L be the subset of RMs (or TMs) that go into an infinite loop. Show that L is undecidable.

Since *L* is the complement of *H*, this seems easy, but we can't fit it neatly into our definition of a mapping reduction.

## Definition

Given a decision problem (D, Q), an *oracle* for Q is a 'magic' RM instruction ORACLE<sub>Q</sub>(i) which, given an encoding of  $d \in D$  in  $R_i$ , sets  $R_i$  to contain 1 iff  $d \in Q$ .

## Definition

Given a decision problem (D, Q), an *oracle* for Q is a 'magic' RM instruction ORACLE<sub>Q</sub>(i) which, given an encoding of  $d \in D$  in  $R_i$ , sets  $R_i$  to contain 1 iff  $d \in Q$ .

Consider RMs augmented with an oracle for halting H, sometimes written  $RM^H$ . We'll return to this.

## Definition

Given a decision problem (D, Q), an *oracle* for Q is a 'magic' RM instruction ORACLE<sub>Q</sub>(i) which, given an encoding of  $d \in D$  in  $R_i$ , sets  $R_i$  to contain 1 iff  $d \in Q$ .

Consider RMs augmented with an oracle for halting H, sometimes written  $RM^H$ . We'll return to this.

If a problem P is decidable, is a machine  $RM^P$  more powerful than a standard RM?

## Definition

Given a decision problem (D, Q), an *oracle* for Q is a 'magic' RM instruction ORACLE<sub>Q</sub>(i) which, given an encoding of  $d \in D$  in  $R_i$ , sets  $R_i$  to contain 1 iff  $d \in Q$ .

Consider RMs augmented with an oracle for halting H, sometimes written  $RM^H$ . We'll return to this.

If a problem *P* is decidable, is a machine *RM<sup>P</sup>* more powerful than a standard RM? **No**. No point in having decidable oracles!

Oracles

Rice's Theorem

## **Turing Reductions**

## Definition

# A *Turing reduction* from $P_1$ to $P_2$ is an RM (or TM) equipped with an Oracle for $P_2$ that solves $P_1$ .

# **Turing Reductions**

#### Definition

A *Turing reduction* from  $P_1$  to  $P_2$  is an RM (or TM) equipped with an Oracle for  $P_2$  that solves  $P_1$ .

Decidability results carry across Turing reductions just as with mapping reductions. But mapping reductions make *finer* distinctions of computing power.

Observe that H is Turing-reducible to L, and thus L is also undecidable.

Oracles 000 Rice's Theorem ●0000

# **Rice's Theorem**

A *property* is a set of RM (or TM) descriptions.

# Rice's Theorem

- A *property* is a set of RM (or TM) descriptions.
- A property is *nontrivial* if it contains some but not all descriptions.

# Rice's Theorem

- A *property* is a set of RM (or TM) descriptions.
- A property is *nontrivial* if it contains some but not all descriptions.
- A property *P* is *semantic* if

$$\mathcal{L}(M_1) = \mathcal{L}(M_2) \Rightarrow (\ulcorner M_1 \urcorner \in P \Leftrightarrow \ulcorner M_2 \urcorner \in P)$$

In other words, it concerns the language and not the particular implementation of the machine.

# Rice's Theorem

- A *property* is a set of RM (or TM) descriptions.
- A property is *nontrivial* if it contains some but not all descriptions.
- A property *P* is *semantic* if

$$\mathcal{L}(M_1) = \mathcal{L}(M_2) \Rightarrow (\ulcorner M_1 \urcorner \in P \Leftrightarrow \ulcorner M_2 \urcorner \in P)$$

In other words, it concerns the language and not the particular implementation of the machine.

**Rice's Theorem** 

All nontrivial semantic properties are undecidable.

Computable Functions	Reductions	Oracles	Rice's Theorem
oo	0000000	000	○●○○○
Proof			

Assume to the contrary that a nontrivial semantic property P is decidable, and it is decided by an RM  $M_P$ . W.l.o.g. a RM  $T_{\emptyset}$  that always rejects is not in P — otherwise we shall proceed with the complement of P instead.

Computable Functions	Reductions	Oracles	Rice's Theorem
oo	0000000	000	○●○○○
Proof			

Assume to the contrary that a nontrivial semantic property *P* is decidable, and it is decided by an RM  $M_P$ . W.l.o.g. a RM  $T_{\emptyset}$  that always rejects is not in *P* — otherwise we shall proceed with the complement of *P* instead. Let *T* be a RM with  $\ulcorner T \urcorner ∈ P$ . We build an  $M_P$  oracle-equipped RM *S* to decide  $A_{\text{RM}}$ .

Computable Functions	Reductions	Oracles	Rice's Theorem
	0000000	000	○●○○○
Proof			

Assume to the contrary that a nontrivial semantic property P is decidable, and it is decided by an RM  $M_P$ . W.l.o.g. a RM  $T_{\emptyset}$  that always rejects is not in P — otherwise we shall proceed with the complement of P instead.

Let T be a RM with  $\[T] \in P$ . We build an  $M_P$  oracle-equipped RM S to decide  $A_{RM}$ .

On input  $\langle M, w \rangle$ :

- **1** Build a RM  $N_{M,w}$  which on input *x*, simulates *M* on *w*. If *M* halts and rejects, it rejects. Otherwise, it simulates *T* on *x*, and accepts if *T* accepts.
- 2 Use  $M_P$  to answer if  $\lceil N_{M,w} \rceil \in P$ .

Computable Functions	Reductions	Oracles	Rice's Theorem
oo	0000000	000	○●○○○
Proof			

Assume to the contrary that a nontrivial semantic property P is decidable, and it is decided by an RM  $M_P$ . W.l.o.g. a RM  $T_{\emptyset}$  that always rejects is not in P – otherwise we shall proceed with the complement of P instead.

Let T be a RM with  $\[T] \in P$ . We build an  $M_P$  oracle-equipped RM S to decide  $A_{RM}$ .

On input  $\langle M, w \rangle$ :

- **1** Build a RM  $N_{M,w}$  which on input *x*, simulates *M* on *w*. If *M* halts and rejects, it rejects. Otherwise, it simulates *T* on *x*, and accepts if *T* accepts.
- 2 Use  $M_P$  to answer if  $\lceil N_{M,w} \rceil \in P$ .

Note the language of  $N_{M,w}$  is  $\mathcal{L}(\mathcal{T})$  if *w* is accepted by *M* and  $\emptyset$  otherwise.

Computable Functions	Reductions	Oracles	Rice's Theorem
	0000000	000	○●○○○
Proof			

Assume to the contrary that a nontrivial semantic property *P* is decidable, and it is decided by an RM  $M_P$ . W.l.o.g. a RM  $T_{\emptyset}$  that always rejects is not in *P* – otherwise we shall proceed with the complement of *P* instead.

Let T be a RM with  $\[T] \in P$ . We build an  $M_P$  oracle-equipped RM S to decide  $A_{RM}$ .

On input  $\langle M, w \rangle$ :

- **1** Build a RM  $N_{M,w}$  which on input *x*, simulates *M* on *w*. If *M* halts and rejects, it rejects. Otherwise, it simulates *T* on *x*, and accepts if *T* accepts.
- **2** Use  $M_P$  to answer if  $\lceil N_{M,w} \rceil \in P$ .

Note the language of  $N_{M,w}$  is  $\mathcal{L}(\mathcal{T})$  if *w* is accepted by *M* and  $\emptyset$  otherwise.

We know  $A_{TM}$  is undecidable, so *P* must also be undecidable.

# Applications of Rice's Theorem

The following are all undecidable by Rice's theorem:

- Whether a language (of an RM/TM) is empty.
- Whether a language (of an RM/TM) is non-empty.
- Whether a language (of an RM/TM) is regular.
- Whether a language (of an RM/TM) is context-free.

#### Note

Sometimes we can prove these properties for particular machines, but it is not decidable in general.

# Wrong applications of Rice's Theorem

Rice's theorem cannot be used for these:

- Whether a TM has less than 7 states.
- Whether a TM has a final state.
- Whether a TM has a start state.

# Wrong applications of Rice's Theorem

Rice's theorem cannot be used for these:

- Whether a TM has less than 7 states.
- Whether a TM has a final state.
- Whether a TM has a start state.

(Note how these are properties of machines, not languages)

- Whether a language (of an RM/TM) is a subset of  $\Sigma^*$ .
- Whether a language of an RM is a language of a TM.

# Wrong applications of Rice's Theorem

Rice's theorem cannot be used for these:

- Whether a TM has less than 7 states.
- Whether a TM has a final state.
- Whether a TM has a start state.

(Note how these are properties of machines, not languages)

- Whether a language (of an RM/TM) is a subset of  $\Sigma^*$ .
- Whether a language of an RM is a language of a TM.

(These properties are trivial).

#### Far-reaching Consequence

We cannot write a program that answers a non-trivial question about the black-box behaviours of programs.

## Next time..

We have developed a theory of **undecidable** problems, and shown how reductions can be used to show more problems are undecidable.

We also saw the daisy cutter of undecidability results, Rice's theorem.

## Next time..

We have developed a theory of **undecidable** problems, and shown how reductions can be used to show more problems are undecidable.

We also saw the daisy cutter of undecidability results, Rice's theorem.

#### Next time

We will address semi-decidable problems. What about machines where we always halt if we accept, but if we do not accept, we may loop forever?