COMP4141 Theory of Computation

Lecture 10 Undecidability, Reducability & Rice's Theorem

Ron van der Meyden

CSE, UNSW

Revision: 2016/04/06

(Credits: D Dill, K Englehardt, M Sipser, W Thomas, T Wilke)

Theorem

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts } w \}$$

is Turing-recognisable but undecidable.

In lecture 9, we discussed recognisability of A_{TM}.

Proof idea for the second part.

Assume to the contrary that H is a decider for A_{TM}. Let D be a TM that uses H as a building block. Its inputs are TM descriptions.

D does the following on input $\langle M \rangle$:
write $\langle M, \langle M \rangle \rangle$ on the tape,
run H which will accept or reject,
output the opposite of H's result.

Running D on $\langle D \rangle$ yields:

$$D(\langle D \rangle) = \begin{cases} \textit{accept} & \text{if } D \text{ does not accept } \langle D \rangle \\ \textit{reject} & \text{if } D \text{ accepts } \langle D \rangle \end{cases}$$

which is a contradiction. □

Definition

Say that $L \in \Sigma^*$ is *co-Turing-recognisable* if $\overline{L} = \Sigma^* \setminus L$ is Turing-recognisable.

Theorem

A language is decidable iff it and its complement are Turing-recognisable.

Proof.

"$\Rightarrow$:" is trivial.

"$\Leftarrow$:" We construct a decider D for L from a recogniser M for L and a recogniser M' for $\overline{L}$ by running both recognisers in parallel on copies of the input word, e.g., by using a 2-tape TM. D accepts when M accepts and D rejects when M' accepts. $\square$

Corollary

$\overline{A_{TM}}$ *is not Turing-recognisable.*

Proof.

by the last 3 theorems. □

Reductions

Proving undecidability or unrecognisability from scratch is often cumbersome.

If we can show that some problem B is at least as hard as a known undecidable problem A then we have a proof that B is undecidable, too.

What does "at least as hard as" mean?

It means we can *reduce* every instance of A to an instance of B, that is, we use a solution to B to solve A.

Example: Halting Problem

Theorem

$H_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that halts on input } w \}$ *is undecidable.*

Proof.

by reduction from A_{TM}. Assume H_{TM} is decidable and then show that a decider D for H_{TM} can be used to build a decider for A_{TM}. The decider D' for A_{TM} works as follows. On input $\langle M, w \rangle$ it runs D. If D rejects, then so does D'. Otherwise, if D accepts, run the recogniser for A_{TM} and return what it returns.
This D' decides A_{TM}, however, we know that A_{TM} is undecidable. Consequently, our assumption that D decides H_{TM} must be wrong. □

Definition

A function $f : \Sigma^* \longrightarrow \Sigma^*$ is *computable* if some TM M, on every input $w \in \Sigma^*$, halts with just $f(w)$ on its tape.

Definition

Language A is *mapping reducible* to language B, written $A \leq_{\mathrm{m}} B$, if there is a computable function $f : \Sigma^* \longrightarrow \Sigma^*$, where for every $w \in \Sigma^*$,

$$w \in A \Leftrightarrow f(w) \in B \ .$$

The function f is called the *reduction* of A to B.

Theorem

If $A \leq_m B$ and B is decidable, then A is decidable.

Corollary

If $A \leq_m B$ and A is undecidable, then B is undecidable.

Review of Recursive Languages

A language is recursive if it is accepted by a TM that always halts.

Theorem

If L is recursive, then $\overline{L}$ is also recursive.
Proof: reverse answer.

Theorem

If L and $\overline{L}$ are both recursively enumerable, then L is recursive.
Proof: run recognisers in parallel on two copies of the input.

Proofs by Reduction

This is one of the most important proof techniques in computer science.

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

Then P_2 is at least as hard as P_1.

Example

(Suppose it is well-known that Kevin cannot lift a car.)

Theorem

Kevin cannot lift a loaded truck (P_2).

Proof.

By reduction from the car-lifting problem (P_1).

Suppose Kevin could lift a loaded truck.

Then, Kevin could solve the car-lifting problem by putting the car in a truck and lifting the truck.

But, it is known that Kevin cannot lift a car. □

Important: Make sure you are doing the reduction in the right direction!

known hard problem $\longrightarrow$ new problem

A wrong proof

Example

Theorem

Kevin cannot lift a feather.

Proof.

By reduction to the car-lifting problem.

We can reduce feather lifting to car-lifting by putting the feather in a car.

It is known that Kevin cannot lift a car.

Therefore, Kevin cannot lift a feather! □

(How's that again?)

Definition

Let $P \subseteq \{ \langle M \rangle \mid M \text{ is a TM} \}$ be a language consisting of TM descriptions.

P is *nontrivial* if it contains some but not all TM descriptions.

P is *semantic* if $L(M_1) = L(M_2) \Rightarrow (\langle M_1 \rangle \in P \Leftrightarrow \langle M_2 \rangle \in P)$.

This is the daisy cutter of undecidability results:

Theorem (Rice's Theorem)

All nontrivial semantic properties are undecidable.

Proof by reduction from A_{TM}.

Assume to the contrary that P is a nontrivial semantic property and the TM R_P decides P. W.l.o.g. a TM $T_\emptyset$ that always rejects (i.e., $L(T_\emptyset) = \emptyset$) is not in P, otherwise we proceed with $\overline{P}$ instead of P. Let T be a TM with $\langle T \rangle \in P$. We build a TM S to decide A_{TM}:

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

1. Build a TM $N_{M,w}$ which, on input x:
   1. Simulate M on w. If it halts and rejects, reject.
   2. Otherwise, simulate T on x. If it accepts, accept.
2. Use R_P to answer: If $\langle N_{M,w} \rangle \in P$ then accept, else reject.

$N_{M,w}$ simulates T if $w \in L(M)$, hence $L(N_{M,w}) = L(T)$ if $w \in L(M)$ and $L(N_{M,w}) = \emptyset$ otherwise.

Therefore $S(\langle M, w \rangle) = \mathit{accept}$ iff $\langle N_{M,w} \rangle \in P$ iff $w \in L(M)$. □

Applications of Rice's theorem

- Whether a language (of a TM) is empty is undecidable.
- Whether a language (of a TM) is non-empty undecidable.
- Whether a language (of a TM) is regular is undecidable.
- Whether a language (of a TM) is context-free is undecidable.

This does not mean we cannot prove any of this for a given TM. Sometimes we can!

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 properties are not properties of languages, but properties of TMs.)

- Whether the language is r.e.
- Whether the language is a subset of Σ^*.

Far-Reaching Consequence

We cannot write a program that reliably answers a non-trivial question about the black-box behaviour of programs (as soon as the programming language used to write the to-be-analysed programs is reasonably expressive).

We may be able to write programs that answer non-trivial questions about programs themselves.