COMP4141 Theory of Computation

Lecture 9 Undecidabilty

Ron van der Meyden

CSE, UNSW

Revision: 2016/04/06

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

Decidability

Definition

A language is called *decidable* if there exists a method—any method at all—to determine whether a given word belongs to that language or not.

Definition

A language is *Turing-decidable* (or *recursive*) if it is accepted by a Turing machine that always halts.

The *Church-Turing thesis* says that decidability and Turing-decidability is the same thing, i.e. any language for which membership is decidable by any means whatsoever can already be decided by means of a Turing machine.

Decidable Problems for Regular Languages

Recall that in lecture 7 we mentioned various decidable problems for regular languages.
These can now be reformulated.

Example

The word problem for DFAs can be made more precise by describing an encoding of DFAs as strings. We call

$$A_{\mathrm{DFA}} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts } w \}$$

the *acceptance problem for DFAs*.

Theorem

A_{DFA} is decidable.

Proof idea.

One TM that decides A_{DFA} would simulate B on w by keeping track of B's current state and of how much of the input word w has been read by writing these data items onto the tape. □

More decidable problems for Regular Languages

$$A_{\mathrm{NFA}} = \{\, \langle B, w \rangle \mid B \text{ is an NFA that accepts } w \,\}$$
$$A_{\mathrm{RE}_\Sigma} = \{\, \langle R, w \rangle \mid R \in \mathrm{RE}_\Sigma \wedge w \in L(R) \,\}$$
$$E_{\mathrm{DFA}} = \{\, \langle B \rangle \mid B \text{ is a DFA and } L(B) = \emptyset \,\}$$
$$EQ_{\mathrm{DFA}} = \{\, \langle A, B \rangle \mid A, B \text{ are DFAs and } L(A) = L(B) \,\}$$

Decidable Problems for CFLs

Theorem

$$A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates } w \}$$

is decidable.

Proof idea.

Convert G to CNF. Enumerate all derivations with up to $2|w| - 1$ steps (or 1 step if $w = \epsilon$). If any of these derivations generate w accept, else reject. $\square$

Closure properties for decidable languages

Theorem

The decidable languages are closed under union.

Proof.

Let L and M be languages that are decided by algorithms A and B respectively. In order to decide their union simply run A and B in parallel on the same given input string until they either accept or reject. The input string is accepted iff either one accepts it, and rejected otherwise. □

Question: are they closed under intersection?
And under complementation?

Recognisability

An *algorithm* or *recipe* for recognizing languages is anything that can be fed a word over the given alphabet, and that depending on its input either "accepts" the input after some time, or "rejects" it, or runs forever without accepting or rejecting its input.
An algorithm is *halting*, or *guaranteed to halt*, if the third possibility does not occur.

Definition

A language is called *semi-decidable* (or *recognizable*) if there exists an algorithm—any algorithm at all—that accepts a given string if and only if the string belongs to that language. In case the string does not belong to the language, the algorithm either rejects it or runs forever.

Turing-Recognisability

Definition

A language is called *recognisable* if there exists any algorithm that accepts a given string if and only if the string belongs to that language. In case the string does not belong to the language, the algorithm either rejects it or runs forever.

Definition

A language is *Turing-recognisable* (or *recursive enumerable*) if it is accepted by a Turing machine.

The *Church-Turing thesis* says that recognisability and Turing-recognisability is the same thing, i.e. any language that can be recognised by any means whatsoever can already be recognised by means of a Turing machine.

Undecidability

Theorem

There are undecidable and there are unrecognisable languages.

Proof.

The set of TMs is enumerable, hence countable. Each TM recognises (or even decides) a single language. The set of languages is uncountable. □

It is instructive to give examples of unrecognisable and of recognisable yet undecidable languages.

Theorem (TM word problem)

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

is Turing-recognisable but undecidable.

Proof idea for the first part.

Construct a *universal TM* U, which, on input $\langle M, w \rangle$ simulates M on w and accepts (rejects) if M enters q_{accept} (q_{reject}).
U recognises A_{TM} but it doesn't decide it. □

—The End—