Decidable Problems

We examine the problems for which there is an algorithm.

Decidable Problems

A ***problem*** asks a yes/no question about some input.

The problem is ***decidable*** if there is a program that *always* halts and *always* answers the question correctly.

Decidable Questions & Recursive Languages

Given a problem, we build a language: take all the instances of the question where the answer is yes and convert the instance to a string.

So, a problem is ***decidable*** if and only if the associated language is recursive.

Encodings

We assume a standard ***encoding*** (or representation) of a machine. E.g., the format you would use to describe it if you wrote a Java program that simulated such a machine.

The actual encoding does not matter, as long as it is fixed and one can easily get between A and its encoding $\langle A \rangle$.

We use $\langle A, B \rangle$ to denote the encoding of pair $\{A, B\}$.

Questions about Regular Languages

It is easy to answer questions about regular languages:

Recursive are:

a) The acceptance problem: $A_{fa} = \{ \langle M, w \rangle : M$ *is FA that accepts* $w \}$.

b) The emptiness problem: $Empty_{fa} = \{ \langle M \rangle : M$ *is FA with empty language* $\}$.

c) The equivalence problem: $EQ_{fa} = \{ \langle A, B \rangle : A$ *and* B *are FAs with* $L(A) = L(B) \}$.

We prove part (c).

Proof that Equivalence is Decidable

We build a TM/program that first checks the input has the correct syntax (encodes some pair of FAs). If not, it rejects.

Then, one approach is to construct the FA for the symmetric difference:

$$\Delta(A, B) = (L(A) - L(B)) \cup (L(B) - L(A))$$

and test $\Delta(A, B)$ for emptiness.

Questions about Context-Free Languages

Many questions about context-free grammars or languages can be readily answered:

Recursive are:

a) The acceptance problem: $A_{cfg} = \{ \langle G, w \rangle : G$ *is a CFG and* w *a string of* $L(G) \}$

b) Any context-free language with grammar G

c) The emptiness problem: $Empty_{cfg} = \{ \langle G \rangle : L(G)$ *is empty* $\}$

We prove part (a).

Proof that Acceptance is Decidable

For example, convert to Chomsky Normal Form. Examine all derivations of length $2|w| - 1$ and conclude. (See also CYK algorithm from earlier.)

Totality for CFG is Not Decidable

Surprisingly, perhaps, testing whether a CFG generates *every* string is hard:
$Total_{cfg} = \{ \langle G \rangle : L(G) = \Sigma^* \}$ is not recursive.

This will be established later.

Configurations

Recall that a ***configuration*** of a machine is a complete record of the machine that tells one the current state and contents of memory.

Specifically, for a 1-tape TM, we write the state where the head is. That is, the configuration is written as

$$t_L \, S \, t_R$$

where t_L is the used tape to the left of the head, S the current state, and t_R the used tape to the right of the head.

Configurations and Loops

For a deterministic machine, if the same configuration recurs, then the machine is stuck in an infinite loop. Hence:

Fact. *If there are at most Q possible configurations of a deterministic machine and it runs for longer than Q, then it is in an infinite loop.*

Computation Strings

A ***computation string*** for machine M accepting string w is the string of configurations from start to finish (separated by some special symbol).

Fact. *It is decidable whether a given string is a computation string or not.*

Check that first configuration is correct and matches the input; check that each configuration follows from previous by the rules of M; and check that final configuration is accepting.

Other Models

Recall that we introduced the Chomsky Hierarchy earlier. Though we do not prove it, the top level corresponds to TMs:

Theorem. *There is an unrestricted grammar for a language if and only if it is r.e.*

Linearly Bounded Automata

A ***linearly bounded automaton*** (LBA) is a 1-tape TM whose head is not allowed to move off the input portion of the tape (and there is a device that tells it where the tape starts and finishes).

Though we do not prove it, it turns out that:

Theorem. *There is a nondeterministic LBA for a language if and only if there is a context-sensitive grammar for it.*

Practice

Show that it is decidable whether an NFA M generates all strings from the alphabet.

Solution to Practice

One approach is to convert M to a DFA. It is easy to see whether a DFA accepts every string: only if every state is accepting.

Summary

A problem is decidable if the associated language is recursive. All problems about FAs and REs are decidable; most problems about CFGs and PDAs are decidable. A computation string is a record of the computation of a machine.