Decidable Languages

CS154
Chris Pollett
Apr 3, 2006.

Outline

- Introduction
- Decidable problems for Regular Languages
- Decidable problems for CFLs

Introduction

- We have shown how it is possible to simulate many different models of computation on a Turing Machine.
- Today we look at what sort of problems can be decided by Turing Machines.
- Recall this is a stronger notion than recognized.
- To decide a language we need to be able to accept if the string is in the language **and** reject if it is not.

DFA Acceptance

- The acceptance problem for DFAs, is the problem of determining if a string is in the language of some DFA.
- Let A_{DFA}={<B,w> | B is a DFA that accepts input string w}.

Theorem A_{DFA} is decidable.

Proof Idea Let M be the TM that does the following:

"On input <B,w>, where B is a DFA and w is a string:

1. Simulate B on w
2. If the simulation ends in an accept state, *accept*. If it ends in a nonaccepting state, *reject*."

NFA Acceptance

- Similarly, we can let A_{NFA}={<N,w> | N is an NFA that accepts input string w}.

Theorem A_{NFA} is decidable.

Proof Let N be the TM that does the following:

"On input <N,w>, where N is a NFA and w is a string:

1. Convert N to an equivalent DFA C using the power set construction.
2. Simulate C on w
3. If the simulation ends in an accept state, *accept*. If it ends in a nonaccepting state, *reject*."

Regular Expression Acceptance

- Let A_{REX}={<R,w> | R is a regular expression that generates string w}.

Theorem A_{REX} is decidable.

Proof Let P be the TM that does the following:

"On input <R,w>, where R is a regular expression and w is a string:

1. Convert R to an equivalent DFA C using the regular expression to NFA conversion algorithm followed by the power set construction.
2. Simulate C on w.
3. If the simulation ends in an accept state, *accept*. If it ends in a nonaccepting state, *reject*."

Emptiness Testing

- Another interesting question about a regular language is whether or not it is empty.
- Supposedly, somebody in the 60's at MIT wrote a very complicated thesis about some class of languages showing all its great properties.
- Later it was shown this class of languages was empty. So the thesis was bogus.
- Let $E_{DFA}=\{<A> \mid A \text{ is a DFA and } L(A) \text{ is empty }\}$.

Theorem E_{DFA} is decidable.

Proof A DFA accepts some string iff reaching an accept state from the start state by traveling along the arrows of the DFA is possible. Let T be the following TM which tests for this:

T= "On input <A> where A is a DFA:

1. Mark the start state of A.
2. Repeat until no new states get marked:
    1. Mark any state that has a transition coming into it from any state that is already marked.
3. If no accept state is marked, *accept;* otherwise, *reject*."

Equality Testing

- Emptiness testing can be used to check if two DFAs, A, B, recognize the same language.
- Let $L(C) = (L(A) \cap \overline{L(B)}) \cup (L(B) \cap \overline{L(A)})$
- Notice L(C) is empty iff L(A) =L(B).
- Let EQ_{DFA}={<A,B> | A and B are DFAs and L(A) = L(B)}.

Theorem EQ_{DFA} is decidable.

Proof Let F be the TM which does the following:

F= "On input <A,B>, where A and B are DFAs.

1. Construct C as described above.
2. Run T of the last slide and accept or reject as it does."

CFG Acceptance

- We now turn to the question of decidability for problems related to context-free languages.
- Let A_{CFG}={<G,w> | G is a CFG that generates string w}.

Theorem A_{CFG} is decidable.

Proof Let S be the following Turing machine:

S= “On input <G,w>, where G is a CFG and w is a string:

1. Convert G to Chomsky Normal Form.
2. Run the CYK algorithm according to G on input w.
3. Accept it this algorithm accepts; reject if it rejects.”