CS310

Decidability

Section 4.1/4.2

November 17, 2010

Decidable? Recognizable?

- { x | x is prime, y is prime, x is a substring of y, $x \in \{0..9\}^+$, $y \in \{0..9\}^+$}
- { x | x is prime, y is prime, x is a *proper* substring of y, $x \in \{0..9\}^+$, $y \in \{0..9\}^+$}
- { y | x is prime, y is prime, x is a proper substring of y, $x \in \{0..9\}^+$, $y \in \{0..9\}^+$}

Decidability

- “the power of algorithms to solve problems.” p 165

- What are the limits of algorithmic solvability?
- How can we tell if two Regular Expressions define the same language?
  - or, can we?
- A language is decidable if some TM decides it

Decidable

- Take a question
  - turn it into a language where answer is yes
    - accept: yes
    - reject: no
  - encode in a string
  - build TM
  - If always halts: decidable!

Decidability

- Acceptance Problem (DFA): Does a given DFA, *B*, accept a given string *w*?

- In terms of languages (because we have defined computation as accept/reject a language):
  - A_{DFA} = { <B, w> | B is a DFA that accepts w }
  - For ALL input pairs <B, w> can a single TM be constructed that will decide <B,w> $\in$ A_{DFA}
    - can we build one TM that will work for all DFAs?
    - is there an ***algorithmic*** way to solve this problem?

Theorem

- A_{DFA} is decidable
  - given <B, w> we can decide if <B, w> $\in A_{DFA}$ or <B, w> $\notin A_{DFA}$

- Proof Idea:
  - Use a TM, M, to simulate B with input w
  - Keep track of current state and current position on the input string
  - Update according to the DFA's δ

Also…

- A_{NFA} and $A_{Regular\ Expression}$ are also decidable
  - why?

Emptiness testing

- Does a finite automata accept any strings at all?
  - E_{DFA} = { <A> | A is a DFA and L(A) = ∅}
- Theorem: E_{DFA} is decidable
- Proof Idea:
  - is it possible to reach an accept state from q_0?

Equivalence testing

- Do two DFAs recognize the same language?
  - EQ_{DFA} = {<A, B> | A and B are DFAs and L(A) = L(B)}
- Theorem: EQ_{DFA} is decidable
  - Proof:

Question

- Can we tell if two Regular Expressions define the same language?
  - why or why not?

CFGs

- A_{CFG} = {<G, w> | G is a CFG that generates w}
- A_{CFG} is decidable
- Could enumerate all strings produced by G: could be infinite, though
- Proof Idea

Equivalence of CFGs

- EQ_{CFG} = {<G, H> | G and H are CFL and L(G) = L(H)}
  - not decidable