Decidable Languages

Computable Problems

- From the Church-Turing Thesis, we can use TMs tell if a problem is computable
  - Can represent problems as languages
  - Formulate problems in terms of testing membership in a language
    - If the language is decidable, the problem is decidable
    - Can simulate all previous topics in TMs

- Example: acceptance problem
  - Test whether a DFA accepts a given string
  - Can be expressed as a language, A_{DFA}
    - A_{DFA} = { <B,w>|B is a DFA that accepts input string w}
  - Problem of testing whether a DFA B accepts an input w is the same as testing whether <B,w> is a member of the language A_{DFA}.

Example: Deterministic FA

- Present a TM M that decides A_{DFA}.
- (Implementation Description) M = “On input <B,w>, where B is a DFA and w is a string:
  - 1. Simulate B on input w.
  - 2. If the simulation ends in an accept state, accept.
    - If it ends in a nonaccepting state, reject.
- TM M exists, A_{DFA} is decidable
  - It is possible to test whether a DFA will accept a given string

Example: Nondeterministic FA

- A_{NFA} = {<B,w> | B is an NFA that accepts input string w}
- Present a TM N that decides A_{NFA}.
  - May make use of TM M from previous example

- N = "On input <C,w>, where C is an NFA and w is a string:
  - 1. Convert NFA C to DFA B
  - 2. Run TM M on <B,w>.
  - 3. If M accepts, accept.
    - Otherwise, reject.

- TM N exists, A_{NFA} is decidable
  - It is possible to test whether a NFA will accept a given string

Example: Regular Expressions

- A_{REX} = {<R,w> | R is a regular expression that generates string w}
- Present a TM P that decides A_{REX}.
  - May make use of TM N from previous example

- P = “On input <R,w>, where R is a regular expression and w is a string:
  - 1. Convert RE R to NFA C
  - 2. Run TM N on <C,w>.
  - 3. If N accepts, accept.
    - Otherwise, reject.

- TM P exists, A_{REX} is decidable
  - It is possible to test whether a regular expression generates a specific string

Example: Emptiness Testing

- All previous problems tested whether an FA accepts a particular string.
  - It is sometimes important to test if an FA accepts anything at all.

- E_{DFA} = { <B> | B is a DFA and L(B) = $\emptyset$}
  - Present a TM T that decides E_{DFA}.

- T = "On input <B>, where B is a DFA:
  - 1. Mark the start state of B.
  - 2. Mark any states that can be directly transitioned from a currently marked state
    - Repeat until no new states are marked.
  - 3. If no marked states are accept states, accept.
    - Otherwise, reject.

- TM T exists, E_{DFA} is decidable
  - It is possible to test whether a DFA accepts no strings
  - Test whether DFA's language is empty

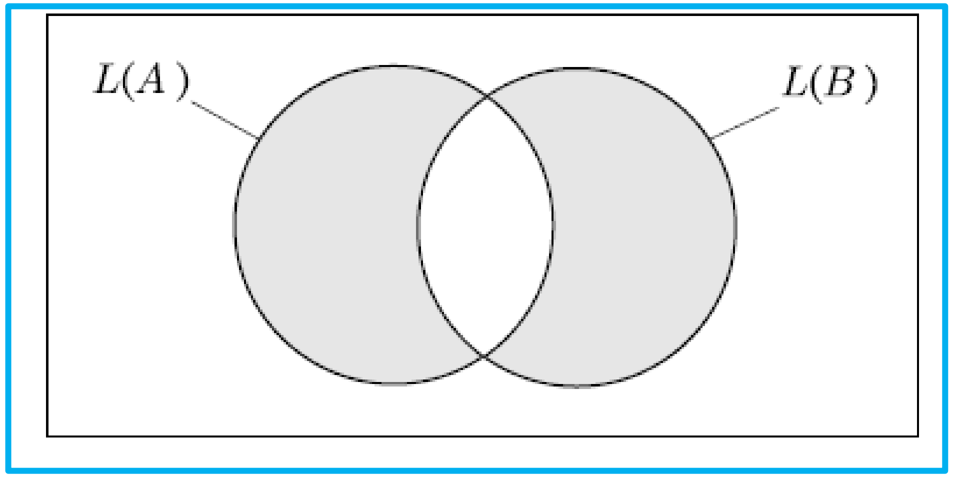
Example: Equivalence Testing

- EQ_{DFA} = { <B_1,B_2> | B_1,B_2 are DFAs and $L(B_1) = L(B_2)$ }
  - Present a TM F that decides EQ_{DFA}.

- Create a DFA C which recognizes strings that are accepted by either B_1 or B_2 but not both.
  - L(C) = symmetric difference
  - For $L(B_1) = L(B_2)$, L(C) must be empty

- F = "On input < B_1,B_2 >, where B_1,B_2 are DFAs:
  - 1. Construct DFA C as described.
  - 2. Run TM T for emptiness testing
  - 3. If T accepts, accept.
    - Otherwise, reject.

- TM F exists, EQ_{DFA} is decidable
  - It is possible to test whether two DFAs are equivalent



Example: Context Free Grammars

- A_{CFG} = { <G,w> | G is a CFG that generates string w}
  - Present a TM S that decides A_{CFG}.

- Systematically produce derivations of G until one matches w
  - May never halt if correct derivation is never encountered
    - TM will be a recognizer but not a decider
  - If rules are put into Chomsky normal form, G is guaranteed to produce string of the correct length within 2n-1 steps
    - n = length of w
    - Only need to check all derivations with 2n-1 steps
      - Finite number of derivations, halting is now guaranteed

Example: Context Free Grammars

- S = “On input < G,w >, where G is a CFG and w is a string:
  - 1. Convert G to equivalent grammar in Chomsky normal form.
  - 2. List all derivations with 2n-1 steps
  - 3. If any derivation generates w, accept
    - Otherwise, reject

- TM S exists, A_{CFG} is decidable
  - It is possible to test if a CFG generates a particular string

Example: CFG Emptiness

- E_{CFG} = { <G> | G is a CFG and L(G) = ∅}
  - Present a TM R that decides E_{CFG}.

- R = “On input < G>, where G is a CFG:
  - 1. Mark all terminal symbols in G
  - 2. Mark any variables that can be substituted with all marked symbols
    - Repeat until no new variables get marked
  - 3. If start variable is not marked, accept.
    - Otherwise, reject.

- TM R exists, E_{CFG} is decidable
  - It is possible to test if a CFG generates any strings

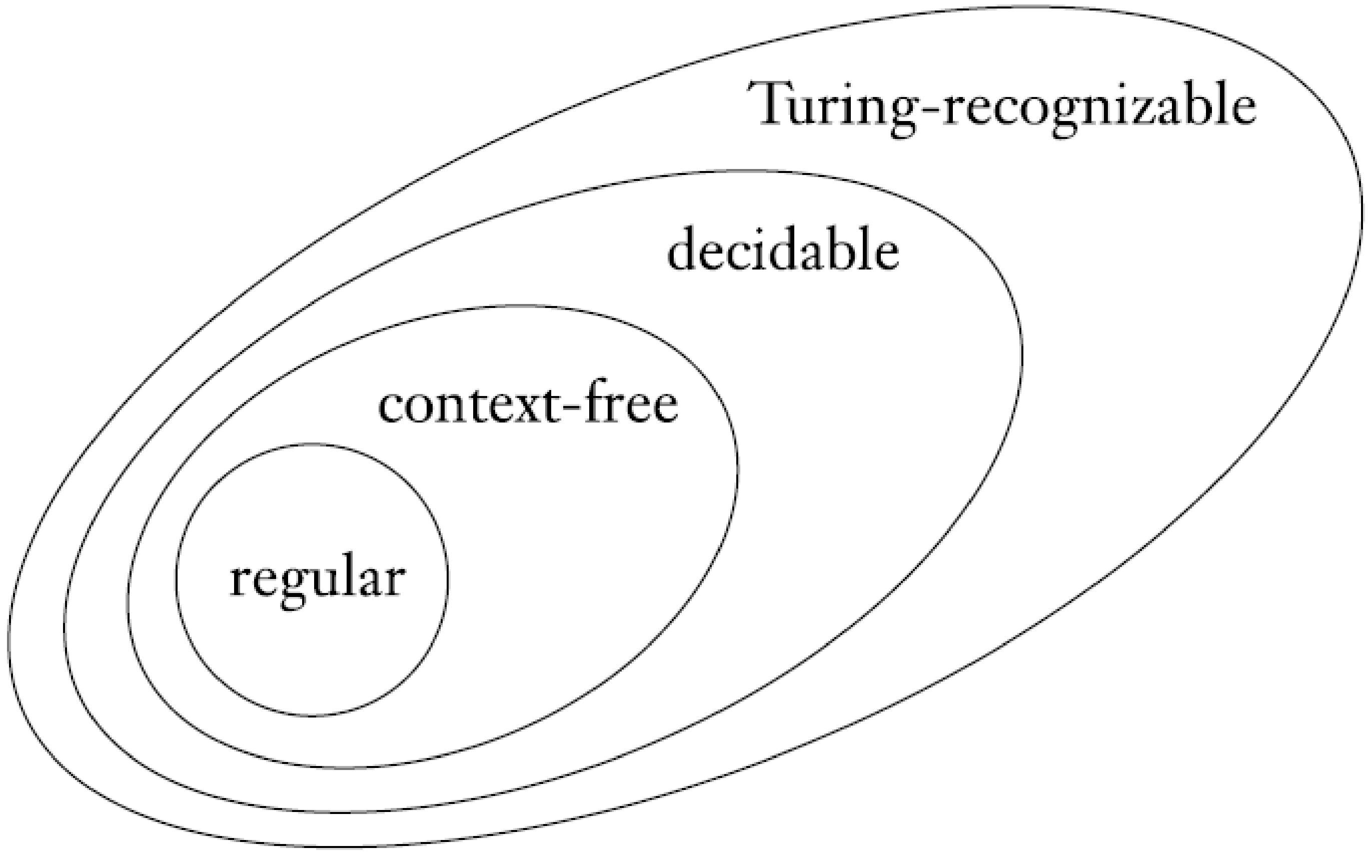


FIGURE 4.10
The relationship among classes of languages

Example: Every CFL is Decidable

- $EQ_{CFG} = \{ \langle G_1, G_2 \rangle \mid G_1, G_2 \text{ are CFGs and } L(G_1) = L(G_2) \}$
  - Present a TM Q that decides EQ_{CFG}.

- Q = “On input < G>, where G is a CFG:
  - 1. Mark all terminal symbols in G
  - 2. Mark any variables that can be substituted with all marked symbols
    - Repeat until no new variables get marked
  - 3. If start variable is not marked, accept.
    - Otherwise, reject.

- TM Q exists, E_{CFG} is decidable
  - It is possible to test if a CFG generates any strings