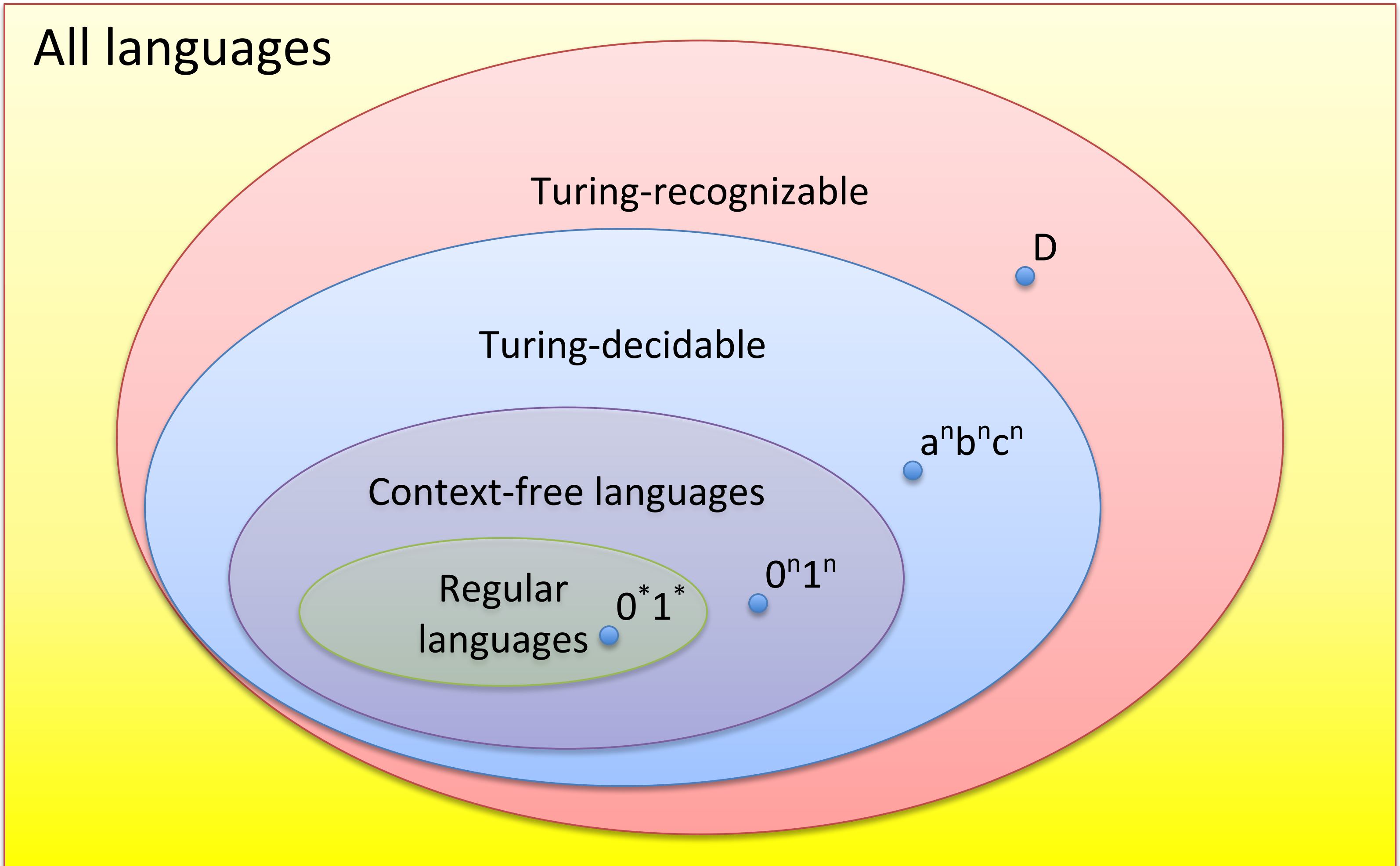
Decidable languages

Sipser 4.1 (pages 165-173)

Hierarchy of languages



Describing Turing machine input

- Input is a string over the alphabet Σ
- What if we want to encode an "object"?
  - DFA, NFA, PDA, CFG, etc…
  - Use brackets shorthand to indicate that input is encoding of object
    - For example:
    - $<M>$ is the encoding of M, where M is a DFA
    - Then, in the machine, simply refer to M as a DFA

Problems concerning regular languages

- $A_{DFA} = \{<B, w>|\ B$ is a DFA that accepts input string $w\}$
- $A_{NFA} = \{<B, w>|\ B$ is a NFA that accepts input string $w\}$
- $A_{REX} = \{<R, w>|$ R is a regular expression that generates string $w\}$

- Are these languages decidable?

Deciding regular languages

- Theorem 4.1: A_{DFA} is decidable.
- Proof.

Let M = "On input $<B,w>$, where B is a DFA:

1. Simulate B on input w.
2. If the simulation ends in an accept state, *accept*. Otherwise, *reject*."

Deciding regular languages

- Theorem 4.1: A_{DFA} is decidable.
- Proof.

Let M = "On input $<B,w>$, where B is a DFA:

1. Simulate B on input w.
2. If the simulation ends in an accept state, *accept*. Otherwise, *reject*."

How do we do this?

What about guessing?

- Theorem 4.2: A_{NFA} is decidable.
- Proof:

  Let N = "On input $<B,w>$, where B is an NFA:

1. Convert NFA B to an equivalent DFA C using the procedure given in Theorem 1.39
2. Simulate TM M of Theorem 4.1 on input $<C,w>$
3. If M accepts, *accept*. Otherwise, *reject*."

Deciding regular expressions

- Theorem 4.3: A_{REX} is decidable.
- Proof:

Let P = "On input $<R,w>$, where R is a regular expression:

1. Convert regular expression R to an equivalent NFA A using the procedure given in Theorem 1.54
2. Simulate TM N of Theorem 4.2 on input $<A,w>$
3. If N accepts, *accept*. Otherwise, *reject*."

Can we test for emptiness?

- $E_{DFA} = \{<A> \mid A$ is a DFA and $L(A) = \emptyset\}$

Can we test for emptiness?

- E_{DFA} = {$<A>$ | A is a DFA and $L(A) = \varnothing$}
- Theorem 4.4: E_{DFA} is a decidable language.
- Proof:

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

1. Mark the start state of A.
2. Repeat until no new states get marked:
   - 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.

Can we tell if two DFAs are equivalent?

- EQ_{DFA} = {<A, B> | A and B are DFAs and $L(A) = L(B)$}

Can we tell if two DFAs are equivalent?

- EQ_{DFA} = {<A, B> | A and B are DFAs and $L(A) = L(B)$}
- Theorem 4.5: EQ_{DFA} is a decidable language.
- Proof:

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

1. Construct DFA C to recognize A XOR B.
2. Run TM T from Theorem 4.4 on input <C>.
3. If T accepts, *accept*. Otherwise, *reject*."

What about context-free languages?

- A_{CFG} = {$<G,w>$| G is a CFG that generates w}

- Theorem 4.7: A_{CFG} is decidable.
- Proof:

  Let S = "On input $<G,w>$, where G is a CFG:

1. Convert G to an equivalent grammar in Chomsky normal form by the procedure given in Theorem 2.9.
2. List all derivations with $2n\text{-}1$ steps, where n is the length of w
3. If any of these derivations generate w, *accept*. Otherwise, *reject*."

Emptiness… again

- E_{CFG} = {$<G>$ | G is a CFG and $L(G)$ = Ø}
- Theorem 4.8: E_{CFG} is decidable.
- Proof:

  Let R = "On input $<G>$, where G is a CFG:

1. Mark all terminal symbols in G.
2. Repeat until no new variables get marked:
   - Mark any variable A where G has a rule $A \rightarrow U_1 U_2 \ldots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked
3. If the start variable is not marked, *accept*; otherwise, *reject*."

Can we tell if two CFGs are equivalent?

- EQ_{CFG} = {$<G, H>$ | G and H are CFGs and $L(G) = L(H)$}
- Is EQ_{DFA} a decidable language?
- Is something wrong with this proof:

  Let W = "On input $<G,H>$, where G and H are CFGs:

  1. Construct CFG F to recognize $L(G)$ XOR $L(H)$.
  2. Run TM R from Theorem 4.8 on input $<F>$.
  3. If R accepts, *accept*. Otherwise, *reject*."