Decidability

Recall: A language L is *decidable* if there exists a TM M such that M recognizes L and M is a decider (always halts).

Encoding Objects as Strings

Many of our examples of computational problems will themselves concern computational devices (such as DFAs or TMs), as well as other structured objects.

- We need a notation for encoding such structured objects as strings.
- For example, a DFA A can be encoded by listing the tuples that define its transition function, with suitable "punctuation":

$$(q_1\#a_1\#b_1)(q_2\#a_2\#b_2)\dots(q_n\#a_n\#b_n)$$

- We want a standard notation that can encode arbitrarily large DFAs using a single tape alphabet. One way to do this is to represent states and input symbols by binary numerals:

  (0000#00#0001)(0001#01#0000)...(1101#00#1000)

  An input to the DFA can similarly be represented:

  $$(00\#01\#10\#00\#00\#11)$$

- We will write $\langle x \rangle$ for the string that encodes object x.

- For decidability theory, the specific details are pretty unimportant, other than that a TM can "understand" the encoding.

- For complexity theory, we will sometimes have to pay closer attention.

Examples of Decidable Problems

The "Acceptance Problem" for DFAs:

$$\mathsf{A}_{\mathsf{DFA}} = \{\langle A, w\rangle \mid A \text{ is a DFA that accepts } w\}$$

Theorem 4.1: $\mathsf{A}_{\mathsf{DFA}}$ is decidable.

Proof: We construct a TM M that operates as follows:

- On input $\langle A, w\rangle$, simulate DFA A on input w. If A accepts, M accepts. If A does not accept, M rejects.

It is important that the simulation of A always terminates, otherwise M would not be a decider.

If the input string to M is not of the form $\langle A, w\rangle$, then M rejects. We will generally not mention this detail.

The "Acceptance Problem" for NFAs:

$$\mathsf{A}_{\mathsf{NFA}} = \{\langle B, w\rangle \mid B \text{ is a NFA that accepts } w\}$$

Theorem 4.2: $\mathsf{A}_{\mathsf{NFA}}$ is decidable.

Proof: *(Book version)* We construct a TM M that operates as follows:

- On input $\langle B, w\rangle$:
  - Convert A to an equivalent DFA A (how?).
  - Run the TM M from Th. 4.1 on input $\langle A, w\rangle$.
  - If M accepts, accept, otherwise reject.

The book points out that “run the TM M from Th. 4.1” implies incorporating M into the design of the present TM.

How else could this theorem have been proved?

The "Acceptance Problem" for Regular Expressions:

$\mathsf{A}_{\mathsf{REX}} = \{\langle R, w\rangle \mid R \text{ is a regular expression with } w \in L(R)\}$

Theorem 4.3: $\mathsf{A}_{\mathsf{REX}}$ is decidable.

Proof: Similar to Th. 4.2.

How else could this be proved?

The "Emptiness Problem" for DFAs:

$$E_{\mathsf{DFA}} = \{\langle A \rangle \mid A \text{ is a DFA with } L(A) = \emptyset\}$$

Theorem 4.4: E_{DFA} is decidable.

Proof: We construct a TM M that operates as follows:

- On input $\langle A \rangle$:
  - Determine the set of states S of A that are reachable from the start state (how?).
  - If any accept state of A is in S, then accept, otherwise reject.

The "Equivalence Problem" for DFAs:

$\mathrm{EQ}_{\mathrm{DFA}} = \{\langle A, B\rangle \mid A$ and B are DFAs with $L(A) = L(B)\}$

Theorem 4.5: $\mathrm{EQ}_{\mathrm{DFA}}$ is decidable.

Proof: Construct a DFA C such that $L(C)$ is the *symmetric difference* of $L(A)$ and $L(B)$. Then run the TM M of Th. 4.4 on C. If M accepts, accept, otherwise reject.

What is the idea for constructing C?

Decidability and Regular Languages

Moral: Pretty much every decision problem you can think of for DFAs and regular languages is decidable.

Decision Problems involving Context-Free Grammars

The "Acceptance Problem" for CFGs:

$$\mathrm{A}_{\mathrm{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates } w\}$$

Theorem 4.7: $\mathrm{A}_{\mathrm{CFG}}$ is decidable.

Proof: On input $\langle G, w\rangle$, convert G to Chomsky normal form, then systematically enumerate all derivations with $2|w| - 1$ steps, where $|w|$ is the length of string w. If any derivation generates w, then accept, otherwise reject.

The point of converting to Chomsky normal form is to be able to bound the length of a derivation of w.

Alternative Proof: On input $\langle G, w \rangle$, convert to Chomsky normal form and then apply the Cocke-Younger-Kasami algorithm to determine if G derives w. The CYK algorithm is much more efficient than a brute-force enumeration of derivations.

The “Emptiness Problem” for CFGs:

$$\mathsf{E}_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG with } L(G) = \emptyset\}$$

Theorem 4.8: $\mathsf{E}_{\mathsf{CFG}}$ is decidable.

Proof: Construct a TM M that implements the following “fixed-point iteration” algorithm:

Stage 0: Mark each terminal symbol of G.

Stage n: If G contains a rule $A \to \gamma$, where all symbols in γ are already marked, then mark A.

Repeat until no new symbols are marked.

If the start symbol of G is not marked, then accept, otherwise reject.

Note that the loop never runs for more than $|V|$ iterations, where $|V|$ is the number of variables (*why?*).

We can prove (*how?*) that symbol A is eventually marked if and only if A derives some sentence.

Prove by induction: For all $n \geq 1$, and all variables A, A derives some sentence via a parse tree of height at most n, if and only if A is marked by stage n.

- Fix $n \geq 1$ and suppose the result has been shown true for all $m < n$.

  Then A derives some sentence via a parse tree of height at most n if and only if there is some rule $A \to \gamma$ such that each variable in γ derives some sentence via a parse tree of height at most m, where $m < n$.

  This is true if and only if every variable in γ is marked by some stage $m < n$.

  This is true if and only if A is marked by stage n.
  (Recall: all terminals are already marked by stage 1).

Theorem 4.9: Every context-free language is decidable.

How is this different than Th. 4.7?

Proof: Suppose $L = L(G)$ for some CFG G. Construct a TM M that operates as follows:

- On input w, run the TM for A_{CFG} on input w. If this computation accepts, then accept, otherwise reject.

Here "run the TM for A_{CFG}" means that this TM is built in as a subroutine of M.

Relationship between Classes of Languages

(from Sipser)

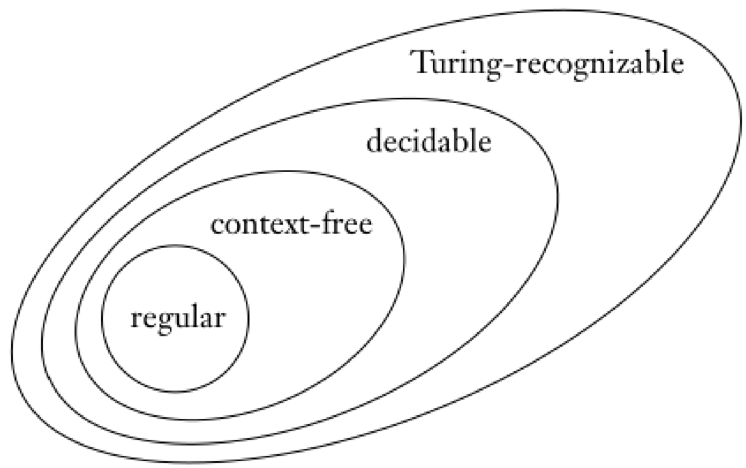

FIGURE 4.10
The relationship among classes of languages

The “Equivalence Problem” for CFGs:

$\mathrm{EQ}_{\mathrm{CFG}} = \{\langle G, H\rangle \mid G$ and H are CFGs with $L(G) = L(H)\}$

Note:

- We can’t determine if $L(G) = L(H)$ by enumerating strings; there are infinitely many.
- We can’t use the symmetric difference trick we used for DFAs, because the class of CFLs is *not* closed under symmetric difference (it is closed under union, but not under intersection and complementation).

In fact, $\mathrm{EQ}_{\mathrm{CFG}}$ is *undecidable* (to be shown later).