CSE 2001: Introduction to Theory of Computation

Fall 2013

Suprakash Datta

datta@cse.yorku.ca

Office: CSEB 3043

Phone: 416-736-2100 ext 77875

Course page: http://www.cse.yorku.ca/course/2001

Chapter 4: Decidability

We are now ready to tackle the question:

What can computers do and what not?

We do this by considering the question:

Which languages are TM-decidable, Turing-recognizable, or neither?

Assuming the Church-Turing thesis, these are fundamental properties of the languages.

Describing TM Programs

Three Levels of Describing algorithms:

- formal (state diagrams, CFGs, et cetera)
- implementation (pseudo-Pascal)
- high-level (coherent and clear English)

Describing input/output format:

TMs allow only strings $\in\Sigma^*$ as input/output.

If our X and Y are of another form (graph, Turing machine, polynomial), then we use $\langle X,Y\rangle$ to denote 'some kind of encoding $\in\Sigma^*$'.

Deciding Regular Languages

The acceptance problem for deterministic finite automata is defined by:

$A_{DFA} = \{ \langle B,w \rangle \mid B \text{ is a DFA that accepts } w \}$

Note that this language deals with all possible DFAs and inputs w, not a specific instance.

Of course, A_{DFA} is a TM-decidable language.

A_{DFA} is Decidable (Thm. 4.1)

Proof: Let the input <B,w> be a DFA with $B=(Q, \Sigma, \delta, q_{start}, F)$ and $w \in \Sigma^*$.

The TM performs the following steps:

1) Check if B and w are 'proper', if not: "reject"
2) Simulate B on w with the help of two pointers: $P_q \in Q$ for the internal state of the DFA, and $P_w \in \{0,1,\ldots,|w|\}$ for the position on the string. While we increase P_w from 0 to $|w|$, we change P_q according to the input letter w_{Pw} and the transition function value $\delta(P_q,w_{Pw})$.
3) If M accepts w: "accept"; otherwise "reject"

Deciding NFA

The acceptance problem for nondeterministic FA
$A_{NFA} = \{ \langle B,w \rangle \mid B$ is an NFA that accepts $w \}$
is a TM decidable language.

Proof: Let the input $\langle B,w \rangle$ be an NFA with
$B=(Q, \Sigma, \delta, q_{start}, F)$ and $w \in \Sigma^*$.
Use our earlier results on finite automata
to transform the NFA B into an equivalent DFA C.
(See Theorem 1.19 how to do this automatically.)
Use the TM of the previous result on $\langle C,w \rangle$.
This can all be done with one big, combined TM.

Regular Expressions

The acceptance problem

A_{REX} = { <R,w> | R is a regular expression
that can generate w }

is a Turing-decidable language.

Proof Theorem 4.3. On input <R,w>:

1. Check if R is a proper regular expression and w a proper string
2. Convert R into a DFA B
3. Run earlier TM for A_{DFA} on <B,w>

Emptiness Testing (Thm. 4.4)

Another problem relating to DFAs is the <u>emptiness problem</u>:

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

How can we decide this language? This language concerns the behavior of the DFA A on all possible strings.

Less obvious than the previous examples.
Idea: check if an accept state of A is reachable from the start state of A.

Proof for DFA-Emptiness

Algorithm for E_{DFA} on input $A=(Q,\Sigma,\delta,q_{start},F)$:

1) If A is not proper DFA: “reject”
2) Mark the start state of A q_{start}
3) Repeat until no new states are marked:
    a) Mark any states that can be δ-reached from any marked state that is already marked
4) If no accept state is marked, “accept”; else “reject”

DFA-Equivalence (Thm. 4.5)

A problem that deals with two DFAs:
$EQ_{DFA} = \{\langle A,B\rangle \mid L(A) = L(B)\}$

Theorem 4.5: EQ_{DFA} is TM-decidable.

Proof: Look at the *symmetric difference* between the two languages: $(L(A) \cap \overline{L}(B)) \cup (\overline{L}(A) \cap L(B))$
Note: "L(A)=L(B)" is equivalent with an empty symmetric difference between L(A) and L(B).
This difference is expressed by standard DFA transformations: union, intersection, complement.

Proof of Theorem 4.5 (cont.)

Algorithm on given $<$**A,B**$>$:

1) If A or B are not proper DFA: “reject”
2) Construct a third DFA C that accepts the language $(L(A) \cap \overline{L}(B)) \cup (\overline{L}(A) \cap L(B))$ (with standard ‘Chapter 1’ transformations).
3) Decide with the TM of the previous theorem whether or not $C \in E_{DFA}$
4) If $C \in E_{DFA}$ then “accept”; If $C \notin E_{DFA}$ then “reject”

Context-Free language problems

Similar languages for context-free grammars:

$A_{CFG} = \{ \langle G,w \rangle \mid G$ is a CFG that generates $w \}$

$E_{CFG} = \{ \langle G \rangle \mid G$ is a CFG with $L(G)=\emptyset \}$

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

The problem with CFGs and PDAs is that they are inherently non-deterministic.

Recall "Chomsky NF"

A context-free grammar $G = (V,\Sigma,R,S)$ is in <u>Chomsky normal form</u> if every rule is of the form

$A \rightarrow BC$ or $A \rightarrow x$

with variables $A \in V$ and $B,C \in V \setminus \{S\}$, and $x \in \Sigma$

For the start variable S we also allow "$S \rightarrow \varepsilon$"

Chomsky NF grammars are easier to analyze.

The derivation $S \Rightarrow^* w$ requires $2|w|-1$ steps (apart from $S \Rightarrow \varepsilon$).

Deciding CFGs (1)

Theorem 4.6: The language
$A_{CFG} = \{ \langle G,w \rangle \mid G \text{ is a CFG that generates } w \}$
is TM-decidable.

Proof: Perform the following algorithm:

1) Check if G and w are proper, if not “reject”
2) Rewrite G to G’ in Chomsky normal form
3) Take care of $w=\varepsilon$ case via $S\to\varepsilon$ check for G’
4) List all G’ derivations of length $2|w|-1$
5) Check if w occurs in this list;
   if so “accept”; if not “reject”

Deciding CFGs (2)

Theorem 4.7: The language
$E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG with } L(G)=\emptyset \}$
is TM-decidable.

Proof: Perform the following algorithm:

1) Check if G is proper, if not "reject"
2) Let $G=(V,\Sigma,R,S)$, define set $T=\Sigma$
3) Repeat $|V|$ times:
   - Check all rules $B \to X_1 \ldots X_k$ in R
   - If $B \notin T$ and $X_1 \ldots X_k \in T^k$ then add B to T
4) If $S \in T$ then "reject", otherwise "accept"

Equality of CFGs

What about the equality language

$EQ_{CFG} = \{ \langle G,H \rangle \mid G$ and H are CFGs with $L(G)=L(H) \}$?

For DFAs we could use the emptiness decision procedure to solve the equality problem.

For CFGs this is not possible… (why?)
… because CFGs are not closed under complementation or intersection.

Later we will see that EQ_{CFG} is not TM-decidable.

Deciding Languages

We now know that the languages:

A_{DFA} = { <B,w> | B is a DFA that accepts w }

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

are TM decidable.

What about the obvious next candidate

A_{TM} = {<M,w> | M is a TM that accepts w }?

Is one TM capable of simulating all other TMs?

Does there exist a Universal TM?

Given a description <M,w> of a TM M and input w, can we simulate M on w?

We can do so via a <u>universal TM</u> U (2-tape):

1) Check if M is a proper TM
   Let M = $(Q,\Sigma,\Gamma,\delta,q_0,q_{accept},q_{reject})$
2) Write down the starting configuration
   $< q_0w >$ on the second tape
3) Repeat until halting configuration is reached:
   - Replace configuration on tape 2 by next configuration according to δ
4) "Accept" if q_{accept} is reached; "reject" if q_{reject}

Next

Towards undecidability:

- **The Halting Problem**
- **Countable and uncountable infinities**
- **Diagonalization arguments**