School of Computing and Information Systems
COMP30026 Models of Computation Tutorial Week 11

Content:

Practice Problems

P11.1 Show that the class of decidable languages is closed under complement. Why can't we use the same argument to show that the class of Turing recognisable languages is closed under complement?

Solution: Let M be a decider for A. We get a decider for A^c simply by swapping the 'reject' and 'accept' states in M.

The construction won't work if all we know about M is that it is a recogniser for A. Namely, M may fail to terminate for some string $w \in A^c$.

P11.2 Show that the class of decidable languages is closed under concatenation.

Solution: We just show the case for concatenation. Let M_A and M_B be deciders for A and B, respectively. We want to construct a decider for $A \circ B$. It will make our task easier if we utilise nondeterminism. We can construct a nondeterministic Turing machine to implement this routine:

> On input w:
> (1) Split w nondeterministically so that $w = xy$.
> (2) Run M_A on x; reject w if M_A rejects x.
> (3) Run M_B on y; reject w if M_B rejects y.
> (4) Otherwise accept w.

This makes good use of the nondeterministic Turing machine's bias towards acceptance.

P11.3 Show that the class of decidable languages is closed under Kleene star.

Solution: Let M be a decider for A. We can construct a decider for A^* in a similar manner to the case for concatenation, using nondeterminism to make the task easier:

> On input w:
> (1) If $w = \epsilon$ accept w, otherwise set a counter $i = 1$
> (2) Split w nondeterministically into i consecutive substrings $x_1, ...x_i$ such that $w = x_1x_2...x_i$
> (3) Run M on each x_i in the split; accept if M accepts all x_i.
> (4) Increment i by one; reject if $i > |w|$, otherwise go to step 2

This Turing machine will accept iff there is a way to split its input w into substrings that are each in A, and so recognises A^*. We can see that it is a decider by reasoning it always halts: for a given i, there are finite ways of splitting a string into i parts; M is a decider and so will always halt for any input; steps (2) and (3) can only run at most $|w|$ times before the machine halts and rejects. So this recogniser halts for any input and is thus a decider for A^*.

P11.4 Show that the problem of whether the language of a DFA is empty, is decidable. That is, show that the language

$$E_{DFA} = \{\langle D \rangle \mid D \text{ is a DFA and } L(D) = \emptyset\}$$

is decidable. *Hint:* write pseudocode for an algorithm which analyses the graph of the DFA, and argue that your algorithm will not run forever on any input DFA $\langle D \rangle$.

Solution: The idea for creating a decider for E_{DFA} is that we find all nodes in the DFA reachable from the start node, and reject iff at least one of those nodes is an accept state.

> On input $\langle D \rangle$:
> (1) Mark the start state of D
> (2) Repeat until no new states are marked:
> Mark state q_n if there is a transition
> from some marked state q_m to q_n
> (3) Reject $\langle D \rangle$ if an accept state of D is marked.
> Otherwise accept $\langle D \rangle$.

For a given DFA, D, every string x corresponds to a path in the DFA's directed graph where vertices are states and directed edges correspond to state transitions. The path starts from the initial state and ends in an accept state if and only if the string x is in $L(D)$. Therefore, there is a string in $L(D)$ if and only if there is a path from the initial state that reaches an accept state. So this is a recogniser for E_{DFA}. We now argue that it is a decider, i.e. always halts. Every time step (2) loops, at least one new state is marked or it terminates. Since $\langle D \rangle$ has finite states this Turing machine cannot run forever on any input, and thus is a decider for E_{DFA}.

P11.5 Show that the problem of whether the language of a CFG is empty, is decidable. That is, show that the language

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

is decidable. *Hint:* write pseudocode for an algorithm which analyses the rules of the CFG, and argue that your algorithm will not run forever on any input CFG $\langle G \rangle$.

Solution: The proof of this is given in lectures.

P11.6 Consider the alphabet $\Sigma = \{0, 1\}$. The set Σ^* consists of all the *finite* bit strings, and the set, while infinite, turns out to be countable. (At first this may seem obvious, since we can use the function $binary : \mathbb{N} \to \Sigma^*$ defined by

$$binary(n) = \text{ the binary representation of } n$$

as enumerator; however, that is not a surjective function, because the legitimate use of leading zeros means there is no unique binary representation of n. For example, both 101 and 00101 denote 5. Instead the idea is to list all binary strings of length 0, then those of length 1, then those of length 2, and so on: $\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, \ldots$.)

Now consider instead the set $\mathcal{B}$ of *infinite* bit strings. Show that $\mathcal{B}$ is much larger than Σ^*. More specifically, use diagonalisation to show that $\mathcal{B}$ is not countable.

Solution: Assume $\mathcal{B}$ is countable.
Then there is some enumeration of $\mathcal{B}$, such as:

Element								
b_1	$\boxed{0}$	0	1	0	1	1	1	...
b_2	1	$\boxed{0}$	1	1	1	0	1	...
b_3	1	0	$\boxed{1}$	1	1	0	1	...
b_4	0	1	0	$\boxed{0}$	1	0	0	...
$\vdots$								

However, the infinite sequence which has

$$i\text{'th bit} = \begin{cases} 0 & \text{if the } i\text{th bit of } b_i \text{ is 1} \\ 1 & \text{if the } i\text{th bit of } b_i \text{ is 0} \end{cases}$$

is a member of $\mathcal{B}$ that is different from each of the b_i in the ith bit, and so is not covered in our enumeration. Hence no enumeration can exist, and $\mathcal{B}$ is uncountable. This should not be surprising, because the set $\mathcal{B}$ is really the same as (or is isomorphic to) $\mathbb{N} \to \Sigma$.

Student Learning Survey

https://unimelb.bluera.com/unimelb/

- What was good? What can be improved?
- Mention tutor/lecturer name specifically
- Mid-Sem Survey Feedback not yet received
- We take feedback seriously. Based on student feedback, we have
  - Removed Haskell
  - Included more resources for discrete math and informal proofs
  - Submitted a handbook change proposal to increase tutorials from 1-hr to 2-hrs
  - Added weekly quizzes to help you catch misunderstandings/misconceptions early
- William:
  - Added illustrations, examples and Python equivalents to make arguments more concrete
  - Added motivation
  - Alternative proof of undecidability