Monash University
Faculty of Information Technology
2nd Semester 2023

FIT2014
Exercises 8
Decidability and Mapping Reductions

ASSESSED PREPARATION: Question 1.
You must provide a serious attempt at this entire question at the start of your class.

1.
(a) How could you encode binary trees over our alphabet {a,b}?

A binary tree has a special vertex called a root. Every non-root vertex has one parent vertex, and every vertex has at most two children: at most one left child and at most one right child. A vertex with no children is called a leaf.

Your encoding scheme should be such that each binary tree can be encoded and different binary trees are encoded differently. Subject to this, it should be reasonably efficient, i.e., not produce strings that are unduly long.

You can ignore any numbering or naming of nodes. We are only interested in representing the abstract structure of the tree.

(b) Give the best upper bound you can for the length of the string used, in your scheme, for binary trees on n nodes.

(c) Outline how a Turing machine would recognise the language of full binary trees.

A binary tree is **full** if every non-leaf node has two children.

(d) How many steps would a Turing machine need to take, in the worst case, to decide if a binary tree on n nodes is full or not?

(e) Now express your time bound in terms of the number ℓ of letters in the string that encodes the tree.

2.

Prove that the following problem is decidable.

Input: two regular expressions R_1 and R_2, using an alphabet Σ.
Question: are the languages described by these expressions disjoint?

3.

Let 2SAT2 be the set of Boolean expressions in CNF, with exactly two literals in each clause (as in 2SAT), such that each variable appears at most twice (which could be as identical literals — both negated, or both unnegated — or could be one negated and one normal).

(a) Prove that every member of 2SAT2 is satisfiable.

(b) How would you work out the number of satisfying truth assignments for such an expression?

4. Consider the mapping reduction from 3-COLOURABILITY to SATISFIABILITY that you did in Assignment 1.

Suppose you only use two colours, say Red and White. How would you change your reduction to make use of this restriction? What is the maximum size of any clause created in the reduction? What mapping reduction are you doing now — in particular, from what language and to what language?

5. Consider the following four problems.

FINITE AUTOMATON ACCEPTANCE
INPUT: Finite Automaton M, string x.
QUESTION: Does M accept x?

NFA ACCEPTANCE
INPUT: Nondeterministic Finite Automaton M, string x.
QUESTION: Does M accept x?

PDA ACCEPTANCE
INPUT: Pushdown Automaton M, string x.
QUESTION: Does M accept x?

TM ACCEPTANCE
INPUT: Turing machine M, string x.
QUESTION: Does M accept x?

(a) Outline a mapping reduction from NFA ACCEPTANCE to FINITE AUTOMATON ACCEPTANCE.

(b) Outline a mapping reduction from FINITE AUTOMATON ACCEPTANCE to TM ACCEPTANCE.

(c) Outline a mapping reduction from PDA ACCEPTANCE to FINITE AUTOMATON ACCEPTANCE.

(d) Given that FINITE AUTOMATON ACCEPTANCE is decidable, what does part (a) tell you about NFA ACCEPTANCE?

(e) Given that FINITE AUTOMATON ACCEPTANCE is decidable, what does part (b) tell you about TM ACCEPTANCE?

6. Consider the following two problems.

CFG GENERATION
INPUT: Context-Free Grammar G, string x.
QUESTION: Does G generate x?

CNF CFG GENERATION
INPUT: Context-Free Grammar G in Chomsky Normal Form, string x.
QUESTION: Does G generate x?

(a) Does there exist a mapping reduction from CFG GENERATION to CNF CFG GENERATION? Why or why not?

(b) Does there exist a mapping reduction from CNF CFG GENERATION to CFG GENERATION? Why or why not?

Supplementary exercises

7.

Suppose you have a Boolean expression φ in Conjunctive Normal Form (CNF), with exactly two literals in each clause. You want to determine whether or not it is in 2SAT.

(a) Suppose some variable appears only in unnegated form (i.e., if the variable is x, then it only appears as x, and never as $\neg x$). How can you eliminate this variable from consideration, and simplify your expression as a result?

(b) Suppose some variable appears only in negated form (i.e., always as $\neg x$, never as x). How can you eliminate this variable from consideration, and simplify your expression as a result?

(c) How can you simplify the expression when some variable appears twice in a single clause?

(d) If there is a clause with just one literal, what can you do about its truth value?

(e) Write an algorithm, Simplify, which takes, as input, a Boolean expression φ in Conjunctive Normal Form (CNF), with **at most** two literals in each clause, and repeatedly applies the actions identified in (a)–(d) for as long as possible. The end result should be *either* rejection, if it is discovered that φ is not satisfiable, *or* it outputs

- a Boolean expression φ' which is satisfiable if and only if φ is satisfiable, and such that in φ': every clause has exactly two literals; every variable appears in both normal and negated form; and that these never appear in the same clause.
- a truth assignment to all variables that appear in φ but not in φ', and which satisfies all clauses of φ that were eliminated in constructing φ'.

(f) Prove, by induction on the number of clauses of φ, that if φ has a clause with just one literal, then Simplify either rejects φ if it is not satisfiable, or outputs a satisfying truth assignment for it and accepts it.

(g) Put it all together to make a polynomial time algorithm for 2SAT.

(h) Would this approach work for 3SAT? Why, or why not?