Theory of Computer Science

G. Röger
Spring Term 2019

University of Basel
Computer Science

Exercise Sheet 9 — Solutions

Exercise 9.1 (Rice's Theorem, 0.5+0.5+0.5+0.5 marks)

For which of the following languages does Rice's theorem show that the language is undecidable? For each language where Rice's theorem can be used, specify the subset of Turing-computable functions $\mathcal{S}$ for which you use the theorem.
Hint: You do not have to write down any proofs. If Rice's theorem is applicable, specify the set $\mathcal{S}$, otherwise give a short reason (1 sentence) why Rice's theorem is not applicable.

(a) $L = \{w \in \{0,1\}^* \mid M_w \text{ computes the binary multiplication function}\}$

(b) $L = \{w \in \{0,1\}^* \mid \text{The output of } M_w \text{ started on the empty tape contains } \texttt{0101}\}$

(c) $L = \{w \in \{0,1\}^* \mid M_w \text{ stops for at least one input after more than 10 steps with a valid output}\}$

(d) $L = \{w \in \{0,1\}^* \mid M_w \text{ computes a binary function over the natural numbers}\}$

Solution:

(a) Rice's theorem is applicable with the set of functions $\mathcal{S} = \{mul\}$.

(b) Rice's theorem is applicable with the following set of functions:

$$\mathcal{S} = \{f \in \mathcal{R} \mid f(\varepsilon) \text{ contains } \texttt{0101}\}$$

(c) Rice's theorem is not directly applicable since the number of steps is a property of the computation and not a property of the computed function.

Note: Without the restriction on the number of steps, i.e., for

$$L' = \{w \in \{0,1\}^* \mid M_w \text{ stops for at least one input with a valid output}\}$$

Rice's theorem would be applicable with the set of functions $\mathcal{S} = \mathcal{R} \setminus \{\Omega\}$. Here, Ω is the function that is undefined everywhere.

(d) Rice's theorem is applicable with the set of functions $\mathcal{S} = \{f \in \mathcal{R} \mid f : \mathbb{N}_0^2 \to \mathbb{N}_0\}$.

Exercise 9.2 (Non-deterministic algorithms, 2.5 marks)

Consider the decision problem VertexCover:

- *Given:* undirected graph $G = \langle V, E \rangle$, number $K \in \mathbb{N}_0$
- *Question:* Does G have a vertex cover of size at most K, i.e., a set of vertices $S \subseteq V$ with $|S| \leq K$ and $\{u, v\} \cap S \neq \emptyset$ for all $\{u, v\} \in E$?

Specify a non-deterministic algorithm for VertexCover, whose runtime is polynomial in the size of the input. Explain why the algorithm is correct and its runtime is polynomial.

Solution:

Input: Undirected Graph $G = \langle V, E \rangle$ and size bound $K \in \mathbb{N}_0$
$V' := V$
$S := \emptyset$
WHILE $|S| \neq K$ and $|S| \neq |V|$:
    GUESS $s \in V'$
    $V' = V' \setminus \{s\}$
    $S = S \cup \{s\}$
FOR $e \in E$:
    IF $e \cap S = \emptyset$:
        REJECT
ACCEPT

The algorithm is correct: We guess a subset $S \subseteq V$ of size $\max\{K, |V|\}$ and check whether it is a vertex cover. Since we can guess every such set, we can guess an actual such vertex cover if there is one. This one passes the test and the algorithm accepts the input. If there is no such vertex cover, the test will fail independently of the guessed set.
Regarding runtime: Guessing the subset takes at most K iterations of the **WHILE** loop. Since the GUESS step and removal from/addition to a set are possible in polynomieller time, guessing the candidate set is possible in polynomial time. Verifying that it is indeed a vertex cover takes $|E|$ tests whether two sets have an empty intersection, which is possible in polynomial time.

Exercise 9.3 (Non-deterministic algorithms, 2.5+3 marks)
Consider the decision problem CLIQUE:

- *Given:* undirected graph $G = \langle V, E \rangle$, number $K \in \mathbb{N}_0$
- *Question:* Does G contain a clique of size K or more, i.e., a set of nodes $C \subseteq V$ with $|C| \geq K$ and $\{u, v\} \in E$ for all $u, v \in C$ with $u \neq v$?

(a) Specify a non-deterministic algorithm for CLIQUE, whose runtime is limited by a polynomial in $|V| + |E|$. Explain why the algorithm's runtime is polynomial.

Solution:

Input: Undirected Graph $G = \langle V, E \rangle$ and size bound $K \in \mathbb{N}_0$
$C = \emptyset$
FOR $v \in V$:
    GUESS $take \in \{0, 1\}$
    IF $take == 1$:
        $C = C \cup \{v\}$
IF $|C| < K$:
    REJECT
FOR $v \in C$:
    FOR $w \in C$:
        IF $v \neq w$ and $\{v, w\} \notin E$:
            REJECT
ACCEPT

The first part can guess every subset of nodes $C \subseteq V$. The second part then verifies that the guessed subset of nodes is a clique. If G contains a clique of size K then these nodes can

be guessed in the first part. The guessed set C then passes all tests and will be accepted. If G does not contain a clique of size K, every choice of C leads to a REJECT, either because C has less than K elements or because there is no edge in G between two elements of C.

Every iteration of the first FOR-loop can be done in constant time (e.g. if we represent C as a list), so the first FOR-loop requires time $O(|V|)$ in total.

The test $|C| < K$ is possible in constant time. Each of the two nested FOR-loops iterate over $|C|$ elements and the test in the inner loop is possible in constant time. Since $|C| \leq |V|$ we get a total running time of $O(|V|^2)$ for the second part.

In total the algorithm runs in time $O(|V| + |V|^2) = O(|V|^2)$, i.e., in polynomial time.

(b) Specify a deterministic algorithm for CLIQUE and analyse its runtime in O-notation.

Solution:

Instead of using GUESS statements, we can systematically try all subsets, for example with the following recursive algorithm:

```
FUNCTION compute-recursively(E, K, C, R) :
    IF R = ∅ : // test whether C is a clique
        IF |C| < K :
            RETURN False
        FOR v ∈ C :
            FOR w ∈ C :
                IF v ≠ w and {v, w} ∉ E :
                    RETURN False
        RETURN True
    ELSE :
        v = smallest element from R wrt. an arbitrary order on the vertices
        If compute-recursively(E, K, C ∪ {v}, R \ {v})
            RETURN True
        If compute-recursively(E, K, C, R \ {v})
            RETURN True
        RETURN False
```

Graph $G = (V, E)$ has a clique of size at least K iff *compute-recursively*$(E, K, \{\}, V)$ returns true. This computation iterates systematically through all subsets of V and test whether they are cliques of size K in G.

Let $n = |V|$ be the number of nodes in G. As in part (a), the runtime of of the test whether a given vertex set is a clique is in $O(n^2)$.

The computation in the ELSE block takes at most $O(n)$ steps (for copying and modifying the sets C and R), if we do not count the time for recursive function calls.

For an input with n nodes we try a total of 2^n subsets and the recursive function calls form a complete binary tree with height n. Such a tree has 2^n leaves and $2^n - 1$ inner nodes. For every inner node the ELSE block is run and for every leaf the IF block is run. Together this means the algorithm runs in time $O((2^n - 1)n + 2^n n^2) = O(n^2 2^n)$.