CS320: Problems and Solutions for Day 12, Winter 2023

Problem 1 Let L be the language defined by the regular expression:

$$(\boldsymbol{bb} \cup \boldsymbol{cc})^*((\boldsymbol{a} \cup \boldsymbol{ba})\boldsymbol{cd})^*$$

(a) Write a complete formal definition of a context-free grammar that generates L. If such grammar does not exist, prove it.

Answer: $G = \{V, \Sigma, P, S\}$, where:
$\Sigma = \{a, b, c, d\}$, $V = \{S, A, B, D\}$,
and the production set P is:

$$\begin{array}{l} S \to AB \\ A \to AA \mid \lambda \mid bb \mid cc \\ B \to BB \mid \lambda \mid Dcd \\ D \to a \mid ba \end{array}$$

(b) Is it possible to write a computer program (algorithm) that operates as follows:

INPUT: An arbitrary Turing machine ζ accepting strings over $\{a, b, c, d\}$.

OUTPUT: **yes** if ζ accepts L and **no** if ζ does not accept L.

Explain your answer briefly.

Answer: No. The property **is equal to L** is a non-trivial property, since (evidently) language L has it, and (for example) the empty set $\emptyset$ does not have it. By Rice's theorem, there is no algorithm that determines if a language accepted by a given Turing machine has a given non-trivial property.

Problem 2 Does there exist an algorithm that solves the following problem:

INPUT: An arbitrary context-free grammar G and an arbitrary compiler $\mathcal{P}$.

QUESTION: Is the language specified by G equal to the set of programs that compile under $\mathcal{P}$?

Explain your answer briefly.

Answer: No. Compiler $\mathcal{P}$ is a program written for a general purpose computer, which is equivalent to a Turing Machine. Hence, the question is equivalent to asking if an arbitrary recursively enumerable language has the property:

is equal to $L(G)$

which is a non-trivial property. By Rice's Theorem, this question does not have an algorithmic answer.

Problem 3 Consider the Turing machine:

$$M = (Q, \Sigma, \Gamma, \delta, p)$$

such that: $Q = \{p, q, s, t, e\}$; $\Sigma = \{a, b, c\}$;
$\Gamma = \{B, X, Y, Z, a, b, c\}$.
and δ is defined by the following transition set:

$$\begin{array}{l} [p, a, q, X, R] \\ [p, b, q, Y, R] \\ [p, c, q, Z, R] \\ \\ [q, a, q, a, R] \\ [q, b, q, b, R] \\ [q, c, q, c, R] \\ [q, B, s, B, L] \\ \\ [s, X, t, B, R] \\ [s, Y, t, B, R] \\ [s, Z, t, B, R] \\ [s, a, e, a, R] \\ [s, b, e, b, R] \\ [s, c, e, c, R] \\ \\ [e, B, e, B, R] \end{array}$$

(where B is the designated blank symbol.)

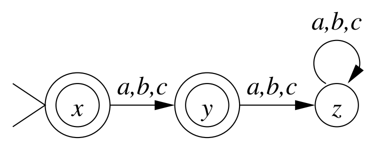
Let L be the set of strings accepted by M (by halting.)

(a) Draw a state-transition graph of a finite automaton M' that accepts L. If such an automaton does not exist, prove it.

Answer: Observe that L is given by the regular expression:

$$\boldsymbol{a} \cup \boldsymbol{b} \cup \boldsymbol{c} \cup \boldsymbol{\lambda}$$

Hence, the following automaton accepts L.



(b) Write a complete formal definition of a Turing machine M_1 that accepts the language L and halts on every input. In short:

$$(\tau \in L) \implies (M_1(\tau) \searrow \text{ and accept })$$

and also:

$$(\tau \notin L) \implies (M_1(\tau) \searrow \text{ and reject })$$

Your construction should be readable as well as accurate; you may comment it. If such a Turing machine does not exist, prove it.

Answer: We employ the algorithmic conversion to obtain M_1 from the deterministic finite automaton constructed in the answer to part (a).

$$M_1 = (Q, \Sigma, \Gamma, \delta, x, F)$$

where: $Q = \{x, y, z\}$; $\Sigma = \{a, b, c\}$;
$\Gamma = \{B, a, b, c\}$; $F = \{x, y\}$.
and δ is defined by the following transition set:

$$\begin{array}{c} [x, a, y, B, R] \\ [x, b, y, B, R] \\ [x, c, y, B, R] \\ \\ [y, a, z, B, R] \\ [y, b, z, B, R] \\ [y, c, z, B, R] \\ \\ [z, a, z, B, R] \\ [z, b, z, B, R] \\ [z, c, z, B, R] \end{array}$$

(where B is the designated blank symbol.)

(c) Write a complete formal definition of a Turing machine M_2 that halts on every input and recognizes Turing machines that accept L. In short:

$$(L(\eta) = L) \implies (M_2(\eta) \searrow \text{ and accept })$$

and also:

$$(L(\eta) \neq L) \implies (M_2(\eta) \searrow \text{ and reject })$$

Your construction should be readable as well as accurate; you may comment it. If such a Turing machine does not exist, prove it.

Answer: Such Turing machine does not exist, since it would decide if an arbitrary Turing machine accepts a language equal to L. The property *"is equal to L"* is non-trivial—hence, by Rice's Theorem it is undecidable whether it holds for the language accepted by an arbitrary Turing machine.