Take-Home Examination and Part 1 Of Mini Project

Concurrency Theory

CCS, Bisimulation, Complexity

Instructions:

- The deadline for Quiz submission is February 22nd AOE. The Submission is to mailed to the instructor as khushraj@iitg.ac.in.
- There is no deadline for the mini project announced yet.
- Take-home exam to be done in groups of three or less.
- Show **all steps** clearly.
- Justify every claim.
- Diagrams are encouraged where appropriate.
- You may use standard definitions from class notes.

Question 1: CCS, LTS Construction, and Behavioral Equivalence

Consider the following CCS processes:

$$P = a.(b.\mathbf{0} + c.\mathbf{0}) + a.b.\mathbf{0}$$
$$Q = a.b.\mathbf{0} + a.(c.\mathbf{0} + b.\mathbf{0})$$

(a) Construct the labelled transition system (LTS) for process P.

(b) Construct the labelled transition system (LTS) for process Q.

(c) Clearly identify all states, transitions, and the initial state in each LTS.

(d) Determine whether P and Q are **strongly bisimilar**.

- If yes, explicitly give a bisimulation relation.
- If not, explain precisely why no such relation exists.

(e) Determine whether P and Q are **weakly bisimilar**.

(f) Briefly explain the difference (if any) between the outcomes for strong and weak bisimulation in this example.

Question 2: Algorithms for Bisimulation and k-Step Bisimulation

Let

$$T_1 = (S_1, \Sigma, \to_1) \quad \text{and} \quad T_2 = (S_2, \Sigma, \to_2)$$

be **finite labelled transition systems**.

A relation $R \subseteq S_1 \times S_2$ is a **bisimulation** if for every $(p, q) \in R$:

- for every transition $p \xrightarrow{a} p'$ in T_1, there exists a transition $q \xrightarrow{a} q'$ in T_2 such that $(p', q') \in R$,
- and symmetrically for transitions from q.

A **k-step bisimulation** is defined similarly, except that single transitions may be matched by paths of length at most k with the same label sequence.

(a) Describe an algorithm to compute the **maximal bisimulation relation** between T_1 and T_2.

(b) Describe an algorithm to compute the **maximal k-step bisimulation relation** between T_1 and T_2.

(c) Explain why both algorithms terminate.

(d) Explain why the relations computed by your algorithms are maximal.

You are not required to provide pseudocode or formal complexity bounds. A clear and precise algorithmic description is sufficient.

Question 3: QBF and Language Inclusion for NFAs

Part A: Quantified Boolean Formulae

A **Quantified Boolean Formula (QBF)** is a formula of the form

$$Q_1 x_1 \, Q_2 x_2 \, \cdots \, Q_n x_n \, . \, \varphi(x_1, \ldots, x_n),$$

where each $Q_i \in \{\forall, \exists\}$ and φ is a propositional formula in **conjunctive normal form (CNF)**.

Decision Problem (QBF-CNF):
Given such a formula, determine whether it is **true** under the standard semantics of alternating quantifiers.

Part B: Language Inclusion for NFAs

A **nondeterministic finite automaton (NFA)** is a tuple $(Q, \Sigma, \delta, q_0, F)$.

Decision Problem (NFA-INCLUSION):
Given two NFAs A and B over the same alphabet Σ, determine whether

$$L(A) \subseteq L(B).$$

Part C: Reductions

(a) Describe a polynomial-time reduction from **QBF-CNF** to **NFA-INCLUSION**.

(b) Describe a polynomial-time reduction from **NFA-INCLUSION** to **QBF-CNF**.

(c) Briefly justify the correctness of both reductions.

A high-level but precise description is sufficient; full constructions are not required.

End of Home Quiz

1 Coding Assignment – Part 1 (5 Marks)

Create a project in **C or Python** that implements the following tasks. The project should be written in a **modular and maintainable manner**, as later stages of the course project may build upon this codebase.

You are encouraged to clearly document your design decisions and provide brief instructions on how to run your code. Do either (Task 1 + Task 2) or (Task 3).

Task 1: CCS Equivalence via Bisimulation

Write a program that:

- Takes two CCS expressions as input,
- Constructs their corresponding labelled transition systems (LTS),
- Checks whether the two CCS processes are equivalent using a **bisimulation algorithm**.

You may assume a basic CCS syntax as covered in the course (prefix, choice, and termination). Your implementation should clearly separate:

- parsing of CCS expressions,
- construction of the LTS,
- bisimulation checking.

Task 2: Maximal Simulation Relation for LTS

Write a program that:

- Takes as input two labelled transition systems,
- Computes and outputs the **maximal simulation relation** between their states.

Your implementation should follow the standard simulation-refinement approach discussed in class and should terminate for finite LTS.

Task 3: Reductions Between QBF and Language Inclusion

Implement the reductions described in **Question 3 of the quiz**:

- Reduction from **QBF (CNF)** to **Language Inclusion for NFAs**,
- Reduction from **Language Inclusion for NFAs** to **QBF (CNF)**.

A high-level but correct implementation is sufficient. You may restrict attention to reasonably small inputs and clearly state any assumptions made.

Note: Correctness, clarity of structure, and extensibility of the code will be considered in evaluation.