Homework #01

Ahmed Echchi
CA-ES-803

Problem 1.1:

* Synchronous/Asynchronous implies blocking/Not blocking but not vice versa. Explain?

=> In a Synchronous communication the sender waits for the receiver's response before continuing its process. Operations are performed in coordinated manner, where each step waits for the previous one to complete.
- The sender's operation is halted until the receiver replies making blocking inherent to synchronous communication.
- However, blocking can occur in other scenarios not strictly synchronous.
- For instance, file I/O or waiting for a lock.
- These ops don't involve a synchronous communication protocol, yet they result in blocking.

=> In an Asynchronous communication by nature, due to presence of the buffer, messages are waitlisted there, transferred to receiver, whenever ready. In the meantime, the sender continues its execution without waiting for acknowledgments, or receiver's reply, ensuring Non-blocking behaviour.
- Nevertheless, Non-blocking operations can occur in synchronous context as well. For example, non-blocking I/O operations within a synchronous framework can allow part of the system to remain non-blocking.
- Multi-threading or event-driven architectures are indeed other means of achieving non-blocking, beyond Async Communication.

Problem 1.2:

Give an example transition system S and an assertion P such that P is always true in S, but is not an invariant of S

Let's consider a transition system S defined by:

$S = \{S_0, S_1, S_2\}$, initial state S_0

transitions: $S_0 \to S_1$; $S_1 \to S_2$; $S_2 \to S_0$

Let P be $P(s): s \in \{S_0, S_1, S_2\}$

=> for the transition system described above P would always hold True.

=> P is not an invariant of S.

By definition, an invariant

is a property that holds at every reachable state, given an initial state and transitions. P does hold for all reachable states. However, it does not express a property that derives from the transitions and initial state - it's simply lists of all possible states.

Problem 1.3:

Let our system be $G=(V,E)$

- First off, we initialize a reachability matrix R of size $|V| \times |V|$, to false. This matrix would store wether node i can reach node j.
- For each $u \in G$, we would perform a traversal algorithm (DFS or BFS). during this traversal we would mark all reachable nodes v from u in reachability Matrix. Specifically R[u][v] would turn true for all nodes v encountered during the traversal.
- We can use Floyd-Warshall algorithm in order to compute the transitive closure of the graph. This would ensure that all indirect paths are also considered, optimizing the reachability matrix.
- The Algo reuses intermediate results of reachability, avoiding redundant computations. Also avoids checking unreachable nodes from any node u as they discarded. Also it scales better with large networks compared to sending individual queries for each pair $(S, \overline{S})$.