A close-up of a logo

Description automatically generated

Modelling Of Software

Intensive Systems

Assignment 2: CBC

1st Master computer science

2024-2025

Liam Leirs

Robbe Teughels

## Reachability / Coverability analysis

For generating the reachability and coverability graphs we used the command ./RC.py [input] [output] [type] as described at the top of the file. For the P-invariants the same command was used but appended with the -p parameter.

Our solution clearly results in an infinite reachability graph, also visible in the coverability graph by the presence of ω in the states. The reason we get an infinite reachability graph is because an unbounded number of new ships can arrive at each tick alongside the fact that the waiting area is unbounded. This allows the system to produce infinitely many reachable states.

To make the reachability graph finite a limit could be placed on the number of allowed tokens in these states (waiting area / arrival). This will only allow a limited number of ships in the system at a time and will thus only generate a finite reachability graph.

**P-invariants analysis**

**M(T\_Generator) = 1:** generator token remains constant ensuring continuous generation of ships. This is to be expected because it keeps the system active.

**M(Berth\_B) + M(Berth\_In\_B) + M(Ship\_Move.Berth\_B\_Cap) + M(Ship\_Move.Berth\_In\_B\_Cap) = 2:** Tokens across the berth B, its input passage, and associated capacity places sum to 2. This models the bounded capacity of the berth and its input passage. This is again to be expected as each berth has a fixed capacity of 1 ship.

**M(Berth\_B) + M(Ship\_Move.Berth\_B\_Cap) = 1:** Ensures that at most 1 ship is either at berth B or its capacity placeholder.

**M(Berth\_A) + M(Ship\_Move.Berth\_A\_Cap) = 1**: Similar to previous one but for berth A.

**M(Passage\_In) + M(Passage\_Out) + 3 \* M(Ship\_Move.Init\_Model) + M(Ship\_Move.Passage\_Cap) = 3:** The passage's total capacity is distributed among incoming, outgoing, and in-transit ships. This models the bounded shared passage. To be expected since the common passage is bounded to 3.

**M(Workers\_Created) = 8**: The total number of workers in the system is fixed at 8.

**M(Berth\_In\_B) + M(Ship\_Move.Berth\_In\_B\_Cap) = 1:** Ensures only 1 ship can occupy berth B’s input passage or its capacity placeholder. This aligns with the uni-directional passage constraint.

**M(Pre\_Passage) = 12:** Represents the initial token count in the pre-passage place, modeling the initial setup.

**M(Berth\_Out\_A) + M(Ship\_Move.Berth\_Out\_A\_Cap) = 1:** At most one ship can occupy berth A’s exit passage or its placeholder. Consistent with uni-directional exit constraints.

**M(Berth\_Out\_B) + M(Ship\_Move.Berth\_Out\_B\_Cap) = 1:** Similar to previous one but for berth B.

**M(Berth\_A) + M(Berth\_B) + M(Berth\_In\_A) + M(Berth\_In\_B) + M(Berth\_Out\_A) + M(Berth\_Out\_B) + ... + 3 \* M(Ship\_Move.Init\_Model) + M(Ship\_Move.Passage\_Cap) = 9:**  Represents global conservation of tokens across all places in the system.

**M(Clock.Workers\_Active) + M(Moving\_Token\_Gen) + ... + M(Work\_B\_Done) = 1:** Represents single clock token that governs the system’s sequential evolution. It ensures proper clock-driven semantics.

**M(Berth\_A) + M(Berth\_In\_A) + M(Passage\_In) + M(Passage\_Out) + ... = 5:** Limits the total number of tokens within berth A’s subsystem and shared passage. Reflects bounded passage capacity.

**M(Post\_Passage) = 28:** Tokens in post-passage place

**M(Berth\_A) + M(Berth\_In\_A) + M(Ship\_Move.Berth\_A\_Cap) + ... = 2:** Total token conservation across berth A and its capacity. Similar to the one for berth B and to be expected.

**M(Berth\_In\_A) + M(Ship\_Move.Berth\_In\_A\_Cap) = 1:** Conservation of tokens in berth A’s input and capacity placeholder.