

A Petri net based Reachability and Safety checking Tool for Open Multi Agent Systems

Shabana AT, Alphy George, S Sheerazuddin, Ajay Kumar, Pranav Prashant and Vipin S Gautam

CSE Department, NIT Calicut, Kozhikode, 673601, Kerala, India

ARTICLE INFO

Keywords:

Formal Verification
Multi-Agent Systems
Multi-Counter Systems
Counter Abstraction

ABSTRACT

We study open multi-agent systems where agents may join or leave the system at runtime. [?] have defined the verification problem for such systems and showed that it is undecidable, in general. Also, they have found one decidable class of open multi-agent systems and provided a partial decision procedure for another. In the same vein, we present a subclass of open MAS, called **Rutwiya** systems, wherein agents communicate exclusively (directly) with the environment and there is no direct communication among the agents. We design an encoding scheme for Rutwiya systems into Petri nets, which are a standard formalism for modeling and analysis of such unbounded distributed systems. This encoding provides a reduction from Rutwiya systems to Petri nets, thereby making reachability (and, therefore, safety checking) in Rutwiya systems decidable. We also develop a tool for reachability in Rutwiya systems that uses any of the already existing Petri net reachability tools, like KReach or ITS, as a back-end to solve the reachability problem in Rutwiya systems.

1. Introduction

Robotic systems represent tangible entities designed to engage with and navigate the physical world. These advanced systems seamlessly combine mechanical, electrical, and computational elements, offering a versatile platform for autonomous or semi-autonomous operations. Instances of robotic systems are found in various applications, such as collaborative robotic MAS in warehouse logistics, where multiple robots work together to optimize inventory management, transportation, and order fulfillment [?]. Another example is the use of swarm robotics in environmental monitoring, where a group of robots collaborates to collect data in large-scale outdoor areas [?].

Robotic systems comprising multiple robotic agents introduce challenges that extend beyond those encountered in the development of single-robot systems. Multi-robot systems can be effectively represented and modelled as multi-agent systems (MAS) [?]. Within the framework of multi-agent systems, each robot is considered an agent capable of sensing its environment, making decisions, and interacting with other agents to accomplish common goals. There are two main types of multi-robot systems: *swarms* [?], in which every robot has the same behaviour, and *MAS*, in which robot can have different behaviours. To achieve sophisticated, global behaviour, a swarm or MAS of robots in a multi-robot system often follow simple, local, nature-inspired protocols to communicate with each other and the environment [?].

Research interests in multi-agent systems (MAS) encompass diverse domains: formal verification to mathematically ensure system correctness [?], distributed decision-making strategies [?], swarm intelligence [?], system learning

integration [?], using agent-based models to simulate and analyze the behaviour of complex systems [?] and so on.

Several techniques have been put forth in the last decade to ensure that multi-agent systems (MAS) fulfill the requirements they are meant to. These approaches encompass verification techniques utilizing both model checking [???] and theorem proving [?]. In traditional methods for modelling and verifying MAS, the number of agents will be fixed at the design time (*closed MAS*) to build and verify models [????]. However, in several real-time applications of MAS, the number of agents cannot be known until runtime. A multi-agent system characterized by agents joining or departing from the system during runtime is referred to as *open MAS (OMAS)* [?]. Prominent instances of OMAS involve scenarios like robotic swarms or MAS with agents capable of departing or experiencing malfunctions, auctions-where participants can dynamically enter or exit the bidding process, and, more broadly, any system where the count of agents is unbounded and fluctuates during runtime.

In [?], they proposed a method for verifying OMAS, which consists of homogeneous agents. They introduced semantics for the analysis of OMAS based on interpreted systems, specifically Open Interpreted Systems (OIS). An OIS models the behavior of agents entering and leaving the system, alongside the environment that represents the rest of the system's state. They showed that given an OIS and a formula ϕ in temporal-epistemic logic, the open model checking problem (OMCP) is undecidable in general. This report aims to extend the semantics of OIS for MAS and describe a fragment called *Rutwiya Systems* for which reachability is shown to be decidable using a counter abstraction based encoding technique.

Observe that agents in open MAS are often simply *replications* of a fixed number of template (finite-state) systems defining the behavior of agents. One way to simplify the system is to take advantage of the regularity that occurs in the induced system model. Counter abstraction can be naturally



shabana_p200052cs@nitc.ac.in (S. AT);

alphy_p200059cs@nitc.ac.in (A. George); sheeraz@nitc.ac.in (S.

Sheerazuddin); ajay_b210459cs@nitc.ac.in (A. Kumar);

pranav_b210460cs@nitc.ac.in (P. Prashant); vipin_b210075cs@nitc.ac.in

(V.S. Gautam)

ORCID(s):

applied in this context. The concept involves representing the global state of an open MAS as a set of counters, with one counter assigned to each local state. These counters keep track of the current count of agents residing in the respective local state. Originally proposed as a method to achieve *symmetry reduction* for fixed-size systems [?], this approach transforms a n -process model, initially exponential in n , into one with a polynomial size in n . We follow the lead of [?] and abstract a subclass of open MAS as Petri nets to show that reachability is decidable for this subclass.

The next sections of the report are organized in the following manner. In Section 2, we provide a review of the syntax and semantics of open MAS. In Section 3, we define a new class of open MAS called Rutwiya Systems, and in Section 4 we show that the reachability of Rutwiya Systems is decidable by encoding it into Petri nets. In Section ??, we discuss the related work. In Section ?? we provide a conclusion and future directions.

2. Open MAS

The system in which agents with different behaviours enter or leave the system at runtime is known as *open MAS*. In this section, we review the syntax and semantics of Open Interpreted System (OIS), proposed by [?]. OIS is based on interpreted systems, originally proposed by [?] to provide computationally grounded semantics to model temporal and epistemic properties of multi-agent and distributed computer systems. Interpreted systems are defined over a set of agents $\{1, 2, \dots, n\}$ with local states for each agent and the environment. A snapshot of the system is represented using a *global state* containing local states of each agent and the environment. Further, the evolution of the system over time can be modelled by the notion of *run*.

2.1. Syntax and Semantics of Open MAS

An agent template defines the behaviour of agents in the open MAS. This is given as follows.

Definition 1 (Agent). Agent template is a tuple $AT = \langle L, \iota, Act, P, tr \rangle$ where:

- $L \neq \emptyset$ is the set of local states of the agents.
- $\iota \in L$ is the unique initial local state of agents.
- Act is the non-empty set of actions the agents execute. Act contains a special symbol τ that corresponds to “no action”.
- $P : L \rightarrow \mathcal{P}(Act)$ is a protocol function that gives the actions enabled at the current state for the agents. Note that $\tau \in P(l)$, for every $l \in L$.
- $tr : L \times Act \times \mathcal{P}(Act) \times Act_E \rightarrow L$ is the local transition function returning the next state of the agent from its current state based on the action executed by that agent, set of actions executed by other agents in the system and the action executed by the environment. Note, that for any $l \in L$, $tr(l, \tau, A, b) = l$, where $A \subseteq Act$ and $b \in Act_E$.

The environment behaviour is defined as follows.

Definition 2 (Environment). The environment is a tuple $E = \langle L_E, \iota_E, Act_E, P_E, tr_E \rangle$ where:

- $L_E \neq \emptyset$ is the set of local states of the environment and $L \cap L_E = \emptyset$.
- $\iota_E \in L_E$ is the unique initial state.
- $Act_E \neq \emptyset$ is a set of actions that the environment executes.
- $P_E : L_E \rightarrow \mathcal{P}(Act_E)$ is a protocol function that defines which actions are enabled at a given local state.
- $tr_E : L_E \times Act_E \times \mathcal{P}(Act) \rightarrow L_E$ is the transition function that gives the next state of the environment given its current state, the action it executes and the set of actions executed by the agents.

We combine the agent templates and environment to obtain open MAS as follows.

Definition 3 (OIS). An open interpreted MAS (OIS) is a $k + 1$ -tuple $\mathcal{O} = \langle AT, E, V \rangle$ where:

- AT is the agent template,
- E is the environment, as defined above, and
- $V : L \rightarrow \mathcal{P}(AP)$ is a labelling function for the agent's local states where AP is a set of atomic propositions.

Now we define the semantics of open MAS. In any instance, the system (open MAS) comprises the environment and zero or more agents. Each agent is assigned an integer – its identity – which is unique to that particular agent. The identity is assigned when the agent joins the system, and it will not change until the agent leaves the system. The environment state along with the agent states and their corresponding identities, at any point of time, gives the snapshot (global state) of the system.

When n agents are present in the system, the set of tuples that characterize it at a particular instance of time is denoted by G_n where G_n is a set of $n + 1$ tuples of the form $(L \times \mathbb{Z}^+) \times \dots \times (L \times \mathbb{Z}^+) \times L_E$ that gives the global state of the system. For a global state g , $g.j$ gives the local state of j^{th} agent and $id(g.j)$ give the identity of j^{th} agent, whereas $g.E$ gives the state of the environment. $G \triangleq \bigcup_{n \in \mathbb{N}} G_n$ is the set of all global states of any size.

For a given $n \in \mathbb{N}$, the set of joint actions $ACT_n = Act^n \times Act_E$. The global transition relation R_n , for a given n , depends on the projection of joint action into a set. A joint action in ACT_n is denoted by a and the projection of joint action into a set as $\hat{a} \triangleq \{a.j \mid a.j \text{ is the action of agent } j \text{ in the joint action } a \text{ and } 1 \leq j \leq n\}$. For a given $i \in [n]$, we define \hat{a}_{-i} – the set of actions in a except the action of agent i – as follows: $\hat{a}_{-i} \triangleq \{a.j \mid 1 \leq j < i\} \cup \{a.j \mid i < j \leq n\}$. Note, \hat{a}_{-i} may include $a.i$.

We now define the temporal evolution of the global states. First, we define the global transition relation of a fixed-size open MAS. Then, we extend the global transition relation for the open MAS with agents leaving and joining the system at runtime.

Definition 4 (The global transition relation of size n).

$R_n \subseteq G_n \times G_n$ on a set G_n of global states is defined by $(g, g') \in R_n$ iff there is some $a \in ACT_n$ such that the following hold:

- $a.E \in P_E(g.E)$ and for all $j \in [n]$, $a.j \in P(g.j)$ where $a.E$ is the environment action and P is the protocol function for the agent.
- $g'.E = tr_E(g.E, a.E, \dot{a})$
- for all $j \in [n]$ it is the case that $id(g'.j) = id(g.j)$ and $g'.j = tr(g.j, a.j, \dot{a}_j, a.E)$, for the j th agent in the system.

The first condition ensures that the action of both environment and agents are enabled as per the protocol functions of the environment and agents, respectively. The second condition guarantees that the transition of the environment is consistent with respect to t_E . The last condition ensures that the agent's identity remains the same after the global transition and each agent's transition is consistent with respect to the corresponding transition relation t_i .

Definition 5 (Global transition relation). The global transition relation $R \subseteq G \times G$ on the set G of global states is defined by $(g, g') \in R$ iff any of the following conditions are met:

Joint action: There is $n \in \mathbb{N}$ such that $(g, g') \in R_n$. This corresponds to a global transition of the system of size n , already defined above.

Agent joining: (i) There is $n \in \mathbb{N}$ such that $g \in G_n$ and $g' \in G_{n+1}$. Also, let \mathbb{D} be the set of agent identities that occur in g . (ii) For all $j \in [n]$ it is the fact that $g.j = g'.j$ and $g.E = g'.E$. (iii) $g'.(n+1) = \iota$ where ι is the initial local state of the agent and $id(g'.(n+1)) \in \mathbb{N} \setminus \mathbb{D}$.

Agent leaving: (i) There is $n \in \mathbb{N}$ such that $g \in G_n$ and $g' \in G_{n-1}$; (ii) There is $\kappa \in [n]$ such that $g' = g_{-\kappa}$, where $g_{-\kappa}$ denotes the tuple resulting from removing the κ th component from the tuple g .

Now, a model of OIS may be defined.

Definition 6 (Model). The associated model of an OIS $\mathcal{O} = \langle AT, E, V \rangle$ is a tuple $O = \langle G, g_0, R, \mathcal{V} \rangle$ where G is the set of global states reachable via R from the initial global state $g_0 = \langle t_E \rangle$ and $\mathcal{V} : G \rightarrow \mathcal{P}(AP \times \mathbb{Z}^+)$ is the labelling function defined for a set of atomic propositions AP as $(p, i) \in \mathcal{V}(g)$ iff there is j with $id(g.j) = i$ and $p \in V(g.j)$. Also, let $Runs(\mathcal{O})$ be the set of all runs of O starting from the initial global state g_0 .

3. Rutwiya Systems

A subclass of open MAS called **Rutwiya Systems** is defined in this section. Later, in Section 4, we show that reachability (and, therefore, safety checking) in this newly defined class is decidable.

In Rutwiya Systems, agents can leave the system at the time of agent leaving transition, but only from a fixed **leaving state**. Further, one or more agents can join in the initial state, at the time of agent joining transition. We can modify the agent template as follows: $AT = \langle L, \iota, Act, P, tr, leave \rangle$ where $leave \in L$ is the leaving state. Also, we insist that $P(leave) = \emptyset$, that is, no action is possible from the leave state. Similarly, for the global transition relation R , the agent joining and agent leaving can be redefined as:

- **Agent joining:** (i) There is $n \in \mathbb{N}$ and $r > 0$ such that $g \in G_n$ and $g' \in G_{n+r}$; (ii) For all $j \in [n]$ it is the case that $g.j = g'.j$. Also $g.E = g'.E$; (iii) For all j such that $1 \leq j \leq r$, $g'.(n+j) = \iota$ where ι is the initial local state of the agent and $id(g'.(n+j))$'s are from the set $\mathbb{N} \setminus \mathbb{D}$, where \mathbb{D} is the set of agent identities occurring in g .
- **Agent leaving:** (i) There is $n \in \mathbb{N}$ and $r > 0$ such that $g \in G_n$ and $g' \in G_{n-r}$; (ii) There is $i \in \mathbb{N}$ such that $g.i = leave$ and $g' = g_{-\{i\}}$, where $g_{-\{i\}}$ denotes the tuple resulting from removing the i^{th} components from g .

Due to the fact that $P(leave) = \emptyset$, when an agent reaches its leave state it is either removed from the system or it continues to remain in the *leave* state. Now, we proceed to show the decidability of reachability (and safety checking) in Rutwiya Systems by encoding them into Petri Nets. Note that we can have more than one leave states in an agent without affecting the decidability of Rutwiya Systems, but we do not consider the case here for the sake of simplicity.

3.1. Inspection Robots - An Example of Rutwiya Systems

Inspection robots play a critical role in various industries, ensuring the safety and integrity of infrastructure in hazardous environments. The need for effective verification and validation of these systems has been highlighted in numerous studies, such as in the work by [?], which outlines the challenges faced by inspection robots in industries ranging from oil and gas to nuclear power. Within this context, the application of Rutwiya Systems becomes particularly relevant.

In a regular inspection scenario, robots must operate with a designated leave state, meaning they can only exit the operational area after reaching this state, ensuring that all necessary inspections are completed. For example, in nuclear waste management, a MAS of heterogeneous robots—including aerial drones, ground-based radiation monitors, and wall-climbing inspectors—must coordinate their actions through implicit communication. This coordination, or joint action, means that each robot's decisions

are influenced by the actions of others. For instance, a ground robot might only initiate radiation scanning if it detects that the aerial drone has signaled the structural integrity of the area. Such scenarios underscore the importance of managing these complex, interdependent actions efficiently, which is where the concept of Rutwiya Systems proves invaluable.

4. Decidability of Rutwiya Systems

We first revisit the notion of Petri nets. A Petri net N is defined formally as a 5-tuple $N = (P, T, F, W, M_0)$ where

- P is a finite set of **places**
- T is a finite set of **transitions**
- F is a **flow relation** over places and transitions, $F \subseteq (P \times T) \cup (T \times P)$,
- W is the **weight function**, $W : F \rightarrow \mathbb{N}$ and
- $M_0 : P \rightarrow \mathbb{N}$ is the **initial marking**.

A firing sequence of Petri net with initial marking M_0 is a sequence of transitions

$$\vec{\sigma} = \langle t_1 \dots t_n \rangle$$

such that there exist a run:

$$M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_{n-1}} M_{n-1} \xrightarrow{t_n} M_n.$$

The set of firing sequence is denoted as $\mathcal{L}(N)$. A marking M is reachable from M_0 if there exist a sequence of firing that transforms M_0 to M . A set of reachable marking for N is denoted as $R(N)$. The reachability graph of N is denoted as G and it consists of nodes as reachable markings and edges as transitions. Also, we define the set of all possible markings for N as follows: $\mathbb{M} = \{M \mid M : P \rightarrow \mathbb{N}\}$

4.1. Encoding of Rutwiya Systems into Petri nets

Given a Rutwiya System $\mathcal{O} = \langle A, E, V \rangle$, we define an encoding of \mathcal{O} as an *interpreted* Petri net $N_{\mathcal{O}} = (P_{\mathcal{O}}, T_{\mathcal{O}}, F_{\mathcal{O}}, W_{\mathcal{O}}, M_{\mathcal{O}}^0, \mathcal{V}_{\mathcal{O}})$ such that **for every run ρ in \mathcal{O} there is a corresponding run ϕ in the encoding $N_{\mathcal{O}}$ and vice versa**. We proceed to describe the various components of $N_{\mathcal{O}}$.

$P_{\mathcal{O}} = L \cup L_E$; For every location in agent template and environment template, it has a corresponding place in $N_{\mathcal{O}}$.

$M_{\mathcal{O}}^0 : P_{\mathcal{O}} \rightarrow \mathbb{N}$ such that $M_{\mathcal{O}}^0(l_E) = 1$ and for all $l \in (P_{\mathcal{O}} - \{l_E\})$, $M_{\mathcal{O}}^0(l) = 0$; Initially there is a token in the place corresponding to l_E , whereas all other places are empty.

We initialize $T_{\mathcal{O}}, F_{\mathcal{O}}, W_{\mathcal{O}}$ as follows:

- $T_{\mathcal{O}} = \{t_{aj}, t_{al}\}$; t_{aj} is the transition in the net corresponding to agent joining action & t_{al} is the transition corresponding to agent leaving action.

- $F_{\mathcal{O}} = \{\langle t_{aj}, l \rangle, \langle leave, t_{al} \rangle\}$; t_{aj} is a source transition that adds tokens to the place l whereas t_{al} is a sink transition that removes tokens from the place $leave$.
- $W_{\mathcal{O}} = \{\langle t_{aj}, l \rangle \mapsto 1, \langle leave, t_{al} \rangle \mapsto 1\}$

Now for every environment transition tuple $\mathfrak{tr} = (l_e, b, A, l'_e)$ in tr_E we augment $T_{\mathcal{O}}, F_{\mathcal{O}}, W_{\mathcal{O}}$ as follows. Corresponding to \mathfrak{tr} , we generate all possible global joint transitions in \mathcal{O} (denoted by $\mathcal{TR}_{\mathfrak{tr}}$) and add them to $T_{\mathcal{O}}$. Further, using the information in $\mathcal{TR}_{\mathfrak{tr}}$, we compute the edge relations in $F_{\mathcal{O}}$ and the weights in $W_{\mathcal{O}}$.

Let $A = \{a_1, a_2, \dots, a_k\}$ be the set of all agent actions executed in a joint action corresponding to the aforementioned environment transition, \mathfrak{tr} . For \mathfrak{tr} we define a set of transition tuples, denoted by $\mathcal{TR}_{\mathfrak{tr}}$, which contains all possible global joint transitions in \mathcal{O} consistent with \mathfrak{tr} .

$\mathcal{TR}_{\mathfrak{tr}}$ contains tuples of the form $\langle (l_1, a_1, A_1, b, l'_1), \dots, (l_n, a_n, A_n, b, l'_n), (l_e, b, A, l'_e) \rangle$ which must satisfy the following property:

- $\{a_1, \dots, a_n\} = A$ and $k \leq n \leq 2 * k$. Also, for any $a \in A$, a occurs at most twice in the sequence $\langle a_1, \dots, a_n \rangle$
- For all $1 \leq i \leq n$, $l'_i = tr(l_i, a_i, A_i, b)$ and $A_i = \{a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n\}$.

Thereafter, we define $T_{\mathfrak{tr}} = \{t_{\mathfrak{tr}} \mid \mathfrak{tr} \in \mathcal{TR}_{\mathfrak{tr}}\}$ as the set of transitions in $N_{\mathcal{O}}$ contributed by \mathfrak{tr} and, therefore, added to $T_{\mathcal{O}}$.

For any $\mathfrak{tr} = \langle (l_1, a_1, A_1, b, l'_1), \dots, (l_n, a_n, A_n, b, l'_n), (l_e, b, A, l'_e) \rangle$ in $\mathcal{TR}_{\mathfrak{tr}}$, let **pre-transition set** $\mathbf{pre}(\mathfrak{tr})$ and **post-transition set** $\mathbf{post}(\mathfrak{tr})$ be defined as:

$$\mathbf{pre}(\mathfrak{tr}) = \{l_1, l_2, \dots, l_n, l_e\}$$

$$\mathbf{post}(\mathfrak{tr}) = \{l'_1, l'_2, \dots, l'_n, l'_e\}$$

where $\mathbf{post}(\mathfrak{tr})$ give the set of all **pre-transition places** and **post-transition places** for the corresponding global joint action \mathfrak{tr} . We add a set $\mathfrak{f}_{\mathfrak{tr}}^{\mathfrak{tr}}$ of (flow) edges to $F_{\mathcal{O}}$ as follows: $\mathfrak{f}_{\mathfrak{tr}}^{\mathfrak{tr}} = \{(l, t_{\mathfrak{tr}}) \mid l \in \mathbf{pre}(\mathfrak{tr})\} \cup \{(t_{\mathfrak{tr}}, l') \mid l' \in \mathbf{post}(\mathfrak{tr})\}$.

Given a global joint action \mathfrak{tr} , for any $l \in \mathbf{pre}(\mathfrak{tr})$ and $l' \in \mathbf{post}(\mathfrak{tr})$, let $n_l^{\mathfrak{tr}}$ and $n_{l'}^{\mathfrak{tr}}$ be the number of times l and l' occurs in the tuples $\langle l_1, l_2, \dots, l_n \rangle$ and $\langle l'_1, l'_2, \dots, l'_n \rangle$, respectively. Then, we may add the following set to weight function $W_{\mathcal{O}}$: $\mathfrak{w}_{\mathfrak{tr}}^{\mathfrak{tr}} = \{(l, t_{\mathfrak{tr}}) \mapsto n_l^{\mathfrak{tr}} \mid l \in \mathbf{pre}(\mathfrak{tr})\} \cup \{(t_{\mathfrak{tr}}, l') \mapsto n_{l'}^{\mathfrak{tr}} \mid l' \in \mathbf{post}(\mathfrak{tr})\}$.

$\mathcal{V}_{\mathcal{O}}$, the *interpretation function*, is the same as V .

We explain the encoding scheme, given above, with an example. Let $\mathcal{O} = \langle A, E \rangle$ be a regular OMAS given as follows:

Agent Template $A = (L, \iota, Act, P, tr, leave)$ with

- $L = \{l_0, l_1, l_2\}$
- $\iota = l_0$
- $Act = \{a_1, a_2\}$
- $P = \{l_0 : a_1, l_1 : a_2\}$
- $leave = l_2$
- $tr = \{\langle l_0, a_1, \{a_1\}, b, l_1 \rangle, \langle l_0, a_1, \emptyset, b, l_1 \rangle, \langle l_1, a_2, \{a_2\}, b, l_2 \rangle, \langle l_1, a_2, \emptyset, b, l_2 \rangle\}$

Environment $E = (L_E, \iota_E, Act_E, P_E, tr_E)$ with

- $L_E = \{l_e\}$
- $\iota_E = l_e$
- $Act_E = \{b\}$
- $P_E = \{l_e : b\}$
- $tr_E = \{\langle l_e, b, \{a_1\}, l_e \rangle, \langle l_e, b, \{a_2\}, l_e \rangle\}$

Then the equivalent Petri net $\langle P_{\mathcal{O}}, T_{\mathcal{O}}, F_{\mathcal{O}}, W_{\mathcal{O}} \rangle$ is given as follows:

- $P_{\mathcal{O}} = L \cup L_E = \{l_0, l_1, l_2, l_e\}$.
- Computation of $T_{\mathcal{O}}$ crucially depends on the computation of $\mathcal{TR}_{\mathcal{O}}$. Note that $\mathcal{TR}_{\mathcal{O}} = \mathcal{TR}_{\langle l_e, b, \{a_1\}, l_e \rangle} \cup \mathcal{TR}_{\langle l_e, b, \{a_2\}, l_e \rangle}$.
 - $\mathcal{TR}_{\langle l_e, b, \{a_1\}, l_e \rangle}$ contains all those global joint transitions which are consistent with environment transition $\langle l_e, b, \{a_1\}, l_e \rangle$ and crucially depend on $(b, \{a_1\})$. Hence, $\mathcal{TR}_{\langle l_e, b, \{a_1\}, l_e \rangle} = \{(\langle l_0, a_1, \emptyset, b, l_1 \rangle, \langle l_e, b, \{a_1\}, l_e \rangle), (\langle l_0, a_1, \{a_1\}, b, l_1 \rangle, \langle l_0, a_1, \{a_1\}, b, l_1 \rangle, \langle l_e, b, \{a_1\}, l_e \rangle)\}$
 - Similarly, $\mathcal{TR}_{\langle l_e, b, \{a_2\}, l_e \rangle}$ contains all those global joint transitions which are consistent with environment transition $\langle l_e, b, \{a_2\}, l_e \rangle$ and crucially depend on $(b, \{a_2\})$. Hence, $\mathcal{TR}_{\langle l_e, b, \{a_2\}, l_e \rangle} = \{(\langle l_1, a_2, \emptyset, b, l_2 \rangle, \langle l_e, b, \{a_2\}, l_e \rangle), (\langle l_1, a_2, \{a_2\}, b, l_2 \rangle, \langle l_1, a_2, \{a_2\}, b, l_2 \rangle, \langle l_e, b, \{a_2\}, l_e \rangle)\}$
- Let us denote the tuples in $\mathcal{TR}_{\langle l_e, b, \{a_1\}, l_e \rangle}$ by \mathbb{t}_1 and \mathbb{t}_2 and $\mathcal{TR}_{\langle l_e, b, \{a_2\}, l_e \rangle}$ by \mathbb{t}_3 and \mathbb{t}_4 , respectively. Then, $T_{\mathcal{O}} = \{t_{aj}, t_{al}, t_{\mathbb{t}_1}, t_{\mathbb{t}_2}, t_{\mathbb{t}_3}, t_{\mathbb{t}_4}\}$.
- The information computed in \mathcal{TR} helps us to augment $F_{\mathcal{O}}$ as follows. Recall that $F_{\mathcal{O}}$ is initialized to be $\{\langle t_{aj}, l_0 \rangle, \langle l_2, t_{al} \rangle\}$. We add the following edges to $F_{\mathcal{O}} : \{\langle l_0, t_{\mathbb{t}_1} \rangle, \langle l_e, t_{\mathbb{t}_1} \rangle, \langle t_{\mathbb{t}_1}, l_1 \rangle\} \cup \{\langle l_0, t_{\mathbb{t}_2} \rangle, \langle l_e, t_{\mathbb{t}_2} \rangle, \langle t_{\mathbb{t}_2}, l_1 \rangle\} \cup \{\langle l_1, t_{\mathbb{t}_3} \rangle, \langle l_e, t_{\mathbb{t}_3} \rangle, \langle t_{\mathbb{t}_3}, l_2 \rangle\} \cup \{\langle l_1, t_{\mathbb{t}_4} \rangle, \langle l_e, t_{\mathbb{t}_4} \rangle, \langle t_{\mathbb{t}_4}, l_2 \rangle\}$.
- The information computed in \mathcal{TR} helps us to compute $W_{\mathcal{O}}$ as follows. Similarly, the initial content of $W_{\mathcal{O}} = \{\langle t_{aj}, l_0 \rangle \mapsto 1, \langle l_2, t_{al} \rangle \mapsto 1\}$. We augment $W_{\mathcal{O}}$

with the following weights computed from \mathcal{TR} , corresponding to the edges in $F_{\mathcal{O}} : \{\langle l_0, t_{\mathbb{t}_1} \rangle \mapsto 1, \langle l_e, t_{\mathbb{t}_1} \rangle \mapsto 1, \langle t_{\mathbb{t}_1}, l_1 \rangle \mapsto 1\} \cup \{\langle l_0, t_{\mathbb{t}_2} \rangle \mapsto 2, \langle l_e, t_{\mathbb{t}_2} \rangle \mapsto 1, \langle t_{\mathbb{t}_2}, l_1 \rangle \mapsto 2\} \cup \{\langle l_1, t_{\mathbb{t}_3} \rangle \mapsto 1, \langle l_e, t_{\mathbb{t}_3} \rangle \mapsto 1, \langle t_{\mathbb{t}_3}, l_2 \rangle \mapsto 1\} \cup \{\langle l_1, t_{\mathbb{t}_4} \rangle \mapsto 2, \langle l_e, t_{\mathbb{t}_4} \rangle \mapsto 1, \langle t_{\mathbb{t}_4}, l_2 \rangle \mapsto 2\}$.

Definition 7. $mf_1 : G \longrightarrow M_{\mathcal{O}}$ such that for any $g \in G$, there is a $M \in \mathbb{M}_{\mathcal{O}}$ where for each $l \in L$, $M(l)$ = number of times l occurs in g and $M(l_e) = 1$.

Definition 8. $mf_2 : M_{\mathcal{O}} \longrightarrow G$ such that for any $M \in \mathbb{M}_{\mathcal{O}}$, there is a $g \in G$ where for each $l \in L$, $M(l)$ = number of times l occurs in g and $l_e = M(l_e) = 1$.

Now, we assert the following claim:

Claim 1. There is a map $\mathfrak{h}_1 : Runs(\mathcal{O}) \longrightarrow Runs(N_{\mathcal{O}})$ such that for every $g \in G$ if there is a run $\rho \in Runs(\mathcal{O})$ from g_0 to g then there is a run $\rho \in \mathfrak{h}_1(\rho)$ from $M_{\mathcal{O}}^0$ to $M = mf_1(g)$.

Proof: Let $\rho = g^0 g^1 g^2 \dots g^i \dots g^n$ be the run of \mathcal{O} , where $g^0 = g_0$ and $g^n = g$. For every $0 \leq i \leq n$, we map g^i to a sequence of configurations (markings) in $N_{\mathcal{O}}$, $\rho^i \in Conf_{\mathcal{O}}^+$, as shown in the Figure 1. The concatenation of all such ρ^i 's gives us a legal run $\rho^0 \cdot \rho^1 \cdot \rho^2 \dots \rho^i \dots \rho^n = \rho \in Runs(N_{\mathcal{O}})$.

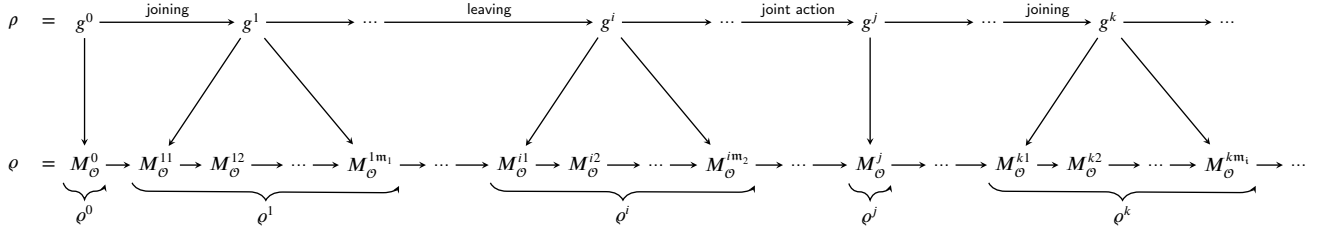
We construct ρ inductively:

$n = 0$: $\rho^0 = M_{\mathcal{O}}^0$ where $M_{\mathcal{O}}(\iota_E) = 1$.

$n > 0$: Assume that ρ^i is well defined for all $i < n$. Suppose $M_{\mathcal{O}}^*$ is the last configuration in the sequence ρ^{n-1} . We need to do the following, for some \mathfrak{m}_n that we compute later: define $\rho^n = M_{\mathcal{O}}^1 M_{\mathcal{O}}^2 \dots M_{\mathcal{O}}^{\mathfrak{m}_n}$ such that $M_{\mathcal{O}}^* \longrightarrow M_{\mathcal{O}}^1$ and for all $j \in \{1, 2, \dots, \mathfrak{m}_n - 1\}$, $M_{\mathcal{O}}^j \longrightarrow M_{\mathcal{O}}^{j+1}$.

As mentioned above, we define ρ^n as a sequence of configurations $M_{\mathcal{O}}^1 M_{\mathcal{O}}^2 \dots M_{\mathcal{O}}^{\mathfrak{m}_n}$. We consider the three cases depending on the type of the transition (g^{n-1}, g^n) and define $M_{\mathcal{O}}^j$ for all $j \in \{1, 2, \dots, \mathfrak{m}_n\}$:

- (g^{n-1}, g^n) is an **agents joining transition**: Let r' be the number of times ι occurs in g^{n-1} and r'' be the number of times ι occurs in g^n . Let $r = r'' - r'$. Let $\mathfrak{m}_n = r$. First we fix $t^j = t_{aj}$, for all $j \in \{1, 2, \dots, \mathfrak{m}_n\}$. Thereafter, we use Algorithm 1 to compute $M_{\mathcal{O}}^j$ for all $j \in \{1, 2, \dots, \mathfrak{m}_n\}$.
- (g^{n-1}, g^n) is an **agents leaving transition**: Let r' be the number of times $leave$ occurs in g^{n-1} and r'' be the number of times $leave$ occurs in g^n . Let $r = r'' - r'$. Let $\mathfrak{m}_n = r$. First, we fix $t^j = t_{al}$, for all $j \in \{1, 2, \dots, \mathfrak{m}_n\}$, as in the case of agent joining. Thereafter, we use Algorithm 2 to compute $M_{\mathcal{O}}^j$, for all $j \in \{1, 2, \dots, \mathfrak{m}_n\}$.

Figure 1: Pictorial representation of construction of ρ from ρ **Algorithm 1:** $M_{\mathcal{O}}^j$ for ρ^n (Agent joining)

```

1  $j \leftarrow 1$ 
2 while  $m_n > 0$  do
3    $M_{\mathcal{O}}^j(l) = M_{\mathcal{O}}^{j-1}(l) + 1$ 
4    $\forall l \in L - \{l\}, M_{\mathcal{O}}^j(l) = M_{\mathcal{O}}^{j-1}(l)$ 
5    $j \leftarrow j + 1$ 
6    $m_n \leftarrow m_n - 1$ 

```

Algorithm 2: $M_{\mathcal{O}}^j$ for ρ^n for ρ^n (Agent Leaving)

```

1  $j = 1$ 
2 while  $m_n > 0$  do
3    $M_{\mathcal{O}}^j(\text{leave}) = M_{\mathcal{O}}^{j-1}(\text{leave}) - 1$ 
4    $\forall l \in L - \{\text{leave}\}, M_{\mathcal{O}}^j(l) = M_{\mathcal{O}}^{j-1}(l)$ 
5    $j \leftarrow j + 1$ 
6    $m_n \leftarrow m_n - 1$ 

```

- (g^{n-1}, g^n) is a **transition with joint action** a^n : Let $g^{n-1}, g^n \in G_z$. That is there are z agents in the system at the time of joint action. Let q be the total number of local states of agents in the system. For every $1 \leq i \leq q$, r_i be the number of times l_i occurs in g^{n-1} and r'_i be the number of times l_i occurs in g^n . Let $r_i = r'_i - r_i$. In the case of joint actions, $m_n = 1$. Algorithm 3 is designed to compute marking $M_{\mathcal{O}}$ from g and g^{n-1} . Now we can select a $\mathbf{t} \in T_{\mathcal{O}}$ such that $\mathbf{t} = \mathbf{t}_{\langle g^{n-1}, i, a^n, i, \dot{a}^n, g^n, i \rangle} \mid i \in [z] \cup \langle g^{n-1}.E, a^n.E, g^n.E \rangle$. Note that, if $a^n.i = \tau$ then there wont be any corresponding element in \mathbf{t} .

Algorithm 3: \mathbf{c} for ρ^n (Joint action)

```

1 for  $i$  in  $q$  do
2   if  $r_i > 0$  then
3      $M_{\mathcal{O}}[l_i] = M_{\mathcal{O}}^*[l_i] + r_i$ 
4   if  $r_i = 0$  then
5      $M_{\mathcal{O}}[l_i] = M_{\mathcal{O}}^*[l_i]$ 
6   if  $r_i < 0$  then
7      $M_{\mathcal{O}}[l_i] = M_{\mathcal{O}}^*[l_i] - r_i$ 

```

The way ρ is constructed, as defined above, makes sure that it is a run in $N_{\mathcal{O}}$.

We complete the proof of Claim 1 by proving the following inductive invariant:

$\forall n : 0 \leq n \leq m, \forall l \in L$, Number of agents in local state l in $g^n =$ Value of $M_{\mathcal{O}}(l)$ in $M_{\mathcal{O}}$.

Proof of invariant: By induction on n

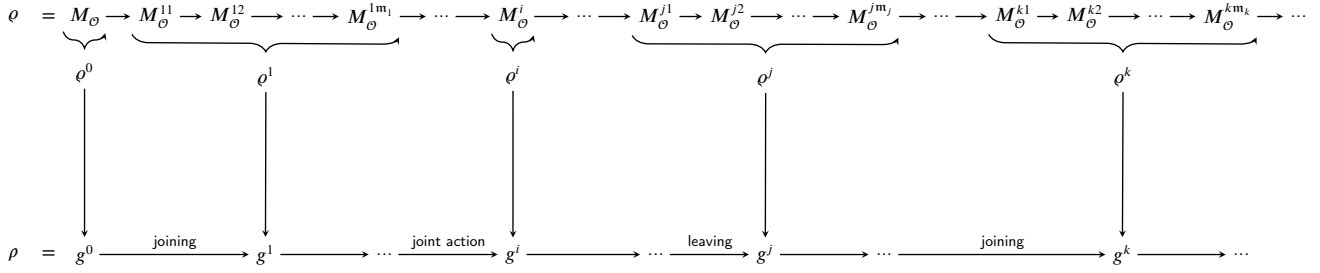
n=0: Trivially holds as $g^0 = \langle l_E \rangle$, i.e., there are no agents in the system initially. Hence, all the number of tokens in $M_{\mathcal{O}}^0$ are set to zero whereas $M_{\mathcal{O}}(l_E) = 1$.

n>0: Inductively, let us assume that invariant holds for $i = n - 1$. We consider the three cases looking at the type of transition (g^{n-1}, g^n) :

agent joining: By inductive hypothesis, $\forall l \in L$, number of agents in local state l in $g^{n-1} =$ number of tokens in $M_{\mathcal{O}}(l)$ in $M_{\mathcal{O}}^{n-1}$. Note that in line number 3 of Algorithm 1, we make sure to increment number of tokens corresponding to l state by this value, while rest of the local state tokens remain unchanged from $M_{\mathcal{O}}^{n-1}$ to $M_{\mathcal{O}}^n$. Therefore, the invariant holds for $i = n$.

agent leaving: By inductive hypothesis, $\forall l \in L$, number of agents in local state l in $g^{n-1} =$ number of tokens in $M_{\mathcal{O}}(l)$ in $M_{\mathcal{O}}^{n-1}$. Note that in line numbers 3 of Algorithm 2, we make sure to decrement tokens corresponding to leave state by this value, while rest of the local state tokens remain unchanged from $M_{\mathcal{O}}^{n-1}$ to $M_{\mathcal{O}}^n$. Therefore, the invariant holds for $i = n$.

joint action: By inductive hypothesis, $\forall l \in L$, number of agents in local state l in $g^{n-1} =$ number of tokens in $M_{\mathcal{O}}(l)$ in $M_{\mathcal{O}}^{n-1}$. Consider an arbitrary $l \in L$, which could be an initial state or leave state. As already mentioned, \mathbf{r} is the difference between the number of times l occurs in g^n and g^{n-1} . Note that this \mathbf{r} will match with the corresponding \mathbf{r} in the definition of joint action in $\Delta_{\mathcal{O}}$. In Algorithm 3, we make sure that $M_{\mathcal{O}}(l_i)$ is decremented \mathbf{r} by if $\mathbf{r} < 0$, and $M_{\mathcal{O}}(l_i)$ is incremented \mathbf{r} by if $\mathbf{r} > 0$, and remains unchanged if $\mathbf{r} = 0$. Therefore, \mathbf{r} is the difference between number of tokens of $M_{\mathcal{O}}(l_i)$ in $M_{\mathcal{O}}^n$ and $M_{\mathcal{O}}^{n-1}$. Hence, the invariant holds for $i = n$.


 Figure 2: Pictorial representation of construction of ρ from ρ

Claim 2. There is a map $\mathfrak{h}_2 : \text{Runs}(N_{\mathcal{O}}) \longrightarrow \text{Runs}(\mathcal{O})$ such that for every $M \in M_{\mathcal{O}}$ if there is a run $\rho \in \text{Runs}(N_{\mathcal{O}})$ from $M_{\mathcal{O}}^0$ to M then there is a run $\rho \in \mathfrak{h}_2(\rho)$ from g_0 to $g = m f_2(M)$.

Proof: Let $\rho = M^0 \xrightarrow{t_0} M^1 \xrightarrow{t_1} M^2 \xrightarrow{t_2} \dots M^i \dots M^m = M$ be a run of $N_{\mathcal{O}}$ where M^0 is the initial marking of $N_{\mathcal{O}}$, where $M^0(l_E) = 1$. For every $0 \leq i \leq m$, we map a sequence of configurations, $\rho^i \in \text{Conf}_{\mathcal{O}}^+$ to g^i as shown in the Figure 2. We can split the run ρ to find such sequences.

Algorithm 4 gives us one particular split $\rho^0, \rho^1, \dots, \rho^i, \dots, \rho^n$, where $n \leq m$. The split is primarily based on the transition t . The sequence of M^i 's where the transition $t = t_{aj}$ are added to ρ^i , for the given i in consideration. Similarly, sequences where $t = t_{al}$ are also added to ρ^i . Otherwise, the algorithm directly adds the M^i to ρ^i .

Algorithm 4: Splitting the run ρ

```

1  $\rho^0 = M_{\mathcal{O}}^0$ 
2  $j \leftarrow 0$ 
3 for  $i$  in  $m$  do
4   if  $t = t_{aj}$  then
5      $j \leftarrow j + 1$ 
6     while  $t = t_{aj}$  do
7        $\rho^j \leftarrow \rho^j \cdot M^i$ 
8        $i \leftarrow i + 1$ 
9   else if  $t = t_{al}$  then
10     $j \leftarrow j + 1$ 
11    while  $t = t_{al}$  do
12       $\rho^j \leftarrow \rho^j \cdot M^i$ 
13       $i \leftarrow i + 1$ 
14   else
15      $\rho^j \leftarrow M^i$ 
16    $i \leftarrow i - 1$ 
```

We construct $\rho = g^0 \cdot g^1 \dots g^i \dots g^n$ from the split $\rho^0, \rho^1, \dots, \rho^i, \dots, \rho^n$ inductively as follows:

$n = 0$: $g^0 = \langle l \rangle$ where $M_{\mathcal{O}}^0(l) = 1$

□

$n > 0$: Assume that g^i is well defined for all $i < n$. For all $0 \leq i < n - 1$, $(g^i, g^{i+1}) \in R$. We need to do the following: define g^n such that $(g^{n-1}, g^n) \in R$.

We define $g^n = \langle (L \times \mathbb{Z}^+), (L \times \mathbb{Z}^+) \dots, (L \times \mathbb{Z}^+), l_E \rangle$ as follows, with $l_E = l$ where $M_{\mathcal{O}}(l) = 1$ and $l \in L_E$. We consider three cases, depending on the type of transition in the first configuration of the sequence ρ^n :

t_{aj} : Let M_p and M_q be the starting and ending configurations of ρ^n . Let $r = M_q(l) - M_p(l)$. Then g^n is constructed from g^{n-1} and ρ^n by adding tuples in the way described in Algorithm 5.

Algorithm 5: g^n for Agent joining

```

1  $\mathbb{D} \leftarrow$  set of all identities present in  $g^{n-1}$ 
2  $id \leftarrow \mathbb{N} \setminus \mathbb{D}$ 
3 for  $j \leftarrow 1, \dots, r$  do
4    $id \leftarrow id + 1$ 
5   add  $\langle t, id \rangle$  to  $g^{n-1}$ 
```

t_{al} : Let M_p and M_q be the starting and ending configurations of ρ^n . Let $r = M_p(\text{leave}) - M_q(\text{leave})$. To construct g^n from g^{n-1} , tuples are removed such that agents in the 'leave' state with the smallest id are removed first. This process is detailed in Algorithm 6, which iterates over each agent type, finds the smallest id for agents that are leaving and removes these tuples from g^{n-1} .

Algorithm 6: g^n for Agent leaving

```

1 for  $j \leftarrow 1, \dots, r$  do
2    $id \leftarrow$  smallest  $id$  among all the  $\text{leave}$  in  $g^{n-1}$ 
3   remove  $\langle \text{leave}, id \rangle$  from  $g^{n-1}$ 
```

t : Let $M_{\mathcal{O}}^n$ be the configuration in ρ^n . Our aim is to construct g^n from $M_{\mathcal{O}}^{n-1}$ and $M_{\mathcal{O}}^n$. We extract the information about joint action from $M_{\mathcal{O}}^n$ and t

and use it to compute the next global state g^n and the label of joint action (a sequence a over Act) for the current state g^{n-1} as described in Algorithm 7.

Algorithm 7: g^n for joint action, given g^{n-1} and t

```

1  $n \leftarrow$  number of agents in  $g^{n-1}$ 
2 Copy  $g^{n-1}$  to  $g^n$ 
3 Initialize  $a$  with  $n$  number of  $\tau$  actions
  //  $a = [\tau, \tau, \dots, \tau]$ 
4  $t_e \leftarrow$  environment tuple from  $t$ 
5 Remove  $t_e$  from  $t$ 
6 Change the state  $t_e[1]$  in  $g^n$  to  $t_e[3]$ 
7 Add  $t_e[2]$  to the  $n + 1^{th}$  position of  $a$ 
8 for each  $t \in t$  do
9   for  $i \leftarrow 1, 2 \dots n$  do
10    if  $g^n.i == t[1]$  then
11       $g^n.i \leftarrow t[4]$ 
12       $a[i] \leftarrow t[2]$ 
13      break
14 return  $g^n, a$ 

```

It is easy to see that ρ as defined above, is a run in \mathcal{O} . \square

Combining the above two claims, we can assert the following:

Lemma 1.

$$Runs(\mathcal{O}) \neq \emptyset \text{ iff } Runs(N_{\mathcal{O}}) \neq \emptyset$$

Theorem 1. *Safety checking in Regular OMAS is decidable.*

Proof: As a proof to the theorem, we reduce the problem of safety checking in Regular Systems to the Liveness ($L1$ – live) problem in Petri net.

A transition t in a Petri net (N, M_0) is said to be $L1$ –live if t occurs at least once in some firing sequence in $\mathcal{L}(M_0)$.

Safety checking problem in Regular OMAS can be defined as follows: Given a Regular OMAS \mathcal{O} , we first define the notion of a global state g being *safe*. Let $g = \langle l_1, l_2, \dots, l_e \rangle$ be a global state in \mathcal{O} , g is safe if $\forall l \in g \setminus l_e, V(l) = \{safe\}$. If there is a location $l \in g \setminus l_e$ such that $V(l) = \{\}$ then g is labeled *unsafe*. A Regular OMAS \mathcal{O} with initial global state g_0 is said to be *safe* if every global state g reachable from g_0 is *safe*. Alternatively, no global state labeled *unsafe* is reachable.

Now the reduction of safety checking to *liveness* is as follows: Recall that a transition t in $N_{\mathcal{O}}$ is in the form $\langle (l_1, a_1, A_1, b, l'_1), (l_2, a_2, A_2, b, l'_2), \dots, (l_e, b, l'_e) \rangle$. First we compute the set of unsafe transitions T_{unsafe} as given in the Algorithm 8. Now for every $t \in T_{unsafe}$, we check whether t is $L1$ – live in $(N_{\mathcal{O}}, M_{\mathcal{O}}^0)$ or not. If any of the t is $L1$ – live, then we conclude that \mathcal{O} is *unsafe*. Otherwise, if all the t are not $L1$ – live then \mathcal{O} is *safe*. \square

Algorithm 8: T_{unsafe} from $T_{\mathcal{O}}$

```

1  $T_{unsafe} = \emptyset$ 
2 for each  $t \in T_{\mathcal{O}}$  do
3   for each  $tp \in t$  do
4     if  $V(tp[5]) == \{\}$  then
5        $T_{unsafe} = T_{unsafe} \cup t$ 
6       Break
7 return  $T_{unsafe}$ 

```

Now, we will see how to check the $L1$ -liveness of a transition t . Let mt be the minimum marking needed to enable the transition t . If mt is reachable from the initial marking, meaning that t occurs at least once in some firing sequence in $\mathcal{L}(M_{\mathcal{O}}^0)$, then t is $L1$ -live. The detailed procedure is provided in Algorithm 9.

Algorithm 9: Safety checking from T_{unsafe} and $M_{\mathcal{O}}^0$

```

1  $\mathcal{M}_{T_{unsafe}} = \emptyset$ 
2 if  $T_{unsafe} = \emptyset$  then
3   return Safe
4 for each  $t \in T_{unsafe}$  do
5    $m_t \leftarrow$  minimum marking needed to enable the
    transition  $t$ 
6    $\mathcal{M}_{T_{unsafe}} \leftarrow \mathcal{M}_{T_{unsafe}} \cup m_t$ 
7 for each  $m_t \in \mathcal{M}_{T_{unsafe}}$  do
8   if  $m_t$  is reachable from  $M_{\mathcal{O}}^0$  then
9     return Unsafe
10 return Safe

```
