Hands-on VITAMIN: A Compositional Tool for Model Checking of Multi-Agent Systems

Angelo Ferrando[1,*,†], Vadim Malvone[2,†]

[1]*University of Modena and Reggio Emilia, Italy*

[2]*Télécom Paris, Institut Polytechnique de Paris, France*

Abstract

Verifying software and hardware systems is challenging due to their complexity, often making exhaustive verification impractical. Transitioning from monolithic systems to Multi-Agent Systems (MAS) exacerbates these challenges, requiring advanced tools for effective verification. Existing tools like MCMAS and STV face limitations in modularity, flexibility, and usability. This paper is based on `VITAMIN` (`VerIficaTion` of `A MultI-ageNt` system), a formal verification framework designed to support various logic and model formalisms while providing a user-friendly experience. Specifically, we apply `VITAMIN` to a case study in the robotics domain, focusing on an extension of Alternating-time Temporal Logic with resource bounds (RB-ATL). Through this case study, we demonstrate `VITAMIN`'s ability to guide end-users in the formal verification process and report experimental results to showcase its usability and feasibility.

Keywords

Model Checking, Multi-Agent Systems, Verification Tools

1. Introduction

Verifying software and hardware systems is challenging due to their complexity and size, often making exhaustive verification impractical without abstraction or optimisation. This complexity demands deep expertise in formal methods, limiting the usability of formal verification techniques in real-world development. One of the most representative approaches to achieve exhaustive formal verification is Model Checking (MC) [1].

Transitioning from monolithic systems to Multi-Agent Systems (MAS) exacerbates these challenges. In fact, as monoloithic systems, MAS face similar verification challenges but with added complexity due to agents' rationality and their interactions. Formal verification for MAS relies heavily on tools like MCMAS [2] and STV [3]. MCMAS is widely used for strategic verification of MAS due to its early development and foundational role in research. However, it has issues hindering broader adoption, including a hard-coded verification process and a lack of modularity, which affects the separation of different logics and models. Despite various extensions, MCMAS lacks transparent extension capabilities for new logics and models, causing maintainability issues. Additionally, its execution can be challenging as it requires additional

WOA 2024: 25th Workshop "From Objects to Agents", July 8-10, 2024, Forte di Bard (AO), Italy

*Corresponding author.

†These authors contributed equally.

angelo.ferrando@unimore.it (A. Ferrando); vadim.malvone@telecom-paris.fr (V. Malvone)

0000-0002-8711-4670 (A. Ferrando); 0000-0001-6138-4229 (V. Malvone)

tools like Eclipse for installation and lacks comprehensive external documentation. These limitations stem from its primary function as a research tool. STV is designed for specific verification goals but lacks the compositional nature and flexibility to support different logics or models for MAS verification. It also lacks comprehensive documentation to assist users and developers. Both MCMAS and STV require a strong background in formal methods, making them challenging for non-expert users. In summary, both tools lack modularity and usability.

These challenges are tackled by VITAMIN (VerIficaTion of A MultI-ageNt system) [4], a new formal verification framework for MAS. VITAMIN aims to be highly compositional, supporting various logic and model formalisms, while also providing a user-friendly experience for both developers and end-users. VITAMIN aims to generalise MAS verification without being tied to specific logic or model formalisms. It achieves compositionality through its design, minimising assumptions about the types of logics and models used. The user experience is enhanced to guide the entire verification process. VITAMIN is still under development, but its compositional nature allows for straightforward extension of its components by external developers.

While [4] discusses the architecture and engineering of VITAMIN, this paper focuses on applying VITAMIN to a interesting case study in the robotics domain. Specifically, we aim to demonstrate how VITAMIN can guide the end-user in the formal verification of a MAS. Although VITAMIN is general and supports various formal logics, to improve readability and better explain its process, we focus on an extension of Alternating-time Temporal Logic (ATL) [5] with resource bounds, called RB-ATL [6]. Using this logic, we present step-by-step how VITAMIN can assist the end-user in verifying an extension of the Curiosity [7] rover case study against formal specifications expressed through RB-ATL. In addition to detailing the steps required by VITAMIN to achieve formal verification, we also report experimental results from the case study. The goal of this work is to demonstrate the high usability of VITAMIN and its feasibility in an application to a robotic case study.

The structure of the paper is as follows: Section 2 introduces all the preliminaries necessary to understand the content of the paper. Section 3 presents the Curiosity rover case study. Section 4 details the step-by-step use of VITAMIN to support the end-user in the formal verification of the case study. Finally, Section 5 concludes the paper and points to future directions.

2. Preliminaries

Let us fix some notation and terminology that will be used in the following. If X is a set and $Y \subseteq X$, we denote by $\overline{Y}$ the complementary set $X \setminus Y$ of Y in X. If π is a sequence, we denote by $|\pi|$ its length and, given $i \leq |\pi|$, we let π_i denote the i-th element of π, $\pi_{\leq i}$ the prefix $\pi_1, \dots, \pi_i$ of π and $\pi_{\geq i}$ the suffix of π starting at π_i. If π is finite, then $last(\pi)$ denote its last element $\pi_{|\pi|}$. If $\alpha = \langle x_1, \dots, x_n \rangle$ is a tuple, then $\alpha[i]$ denotes its i-th component x_i.

Definition 1. *A Concurrent Game Structure (CGS for short) is a tuple* $\mathfrak{G} = \langle Ap, Ag, S, s_I, \{act_i\}_{i \in Ag}, P, t \rangle$ *such that:*

- Ap *is a non-empty set of atomic propositions;*
- $Ag = \{1, \dots, n\}$ *is a finite set of agents;*

- *S is a non-empty set of states and $s_I \in S$ is the initial state;*
- *for any $i \in Ag$, act_i is a set of actions, $ACT = \Pi_{i \in Ag} act_i$ is the set of tuples of actions, and $act = \bigcup_{i \in Ag} act_i$ the set of all actions;*
- *$P : Ag \times S \to (2^{act} \setminus \emptyset)$ is the protocol function that associates to any agent i and state s a non-empty subset of act_i representing the actions that are available for i at s. We impose that the idle action $\star$ always belong to $P(i, s)$ for any i;*
- *$t : S \times ACT \to S$ is the transition function, that is given a state s and a tuple of actions $\mathbf{a}$ (where $\forall i$, $\mathbf{a}[i] \in P(i, s)$) such function outputs a state s';*
- *$L : S \to 2^{Ap}$ is the labeling function associating to any state s a set of atomic propositions; such set can be empty and represents the set of proposition that are true at s.*

If C is a coalition (i.e., a set of agents) and s is a state a **C-action** available at s is a tuple α whose length is $|Ag|$ and such that $\alpha_i \in P(s, i)$ for each $i \in C$ and for each $j \in \overline{C}$, $\alpha_j = \#_j$, where $\#_j$ is a fixed symbol used as placeholder for an arbitrary action of player j. We denote by $Act(C, s)$ the set of all C-actions at s. If $\alpha \in Act(C, s)$ and $\beta \in Act(\overline{C}, s)$ then $\alpha \cdot \beta$ denotes the unique joint action $\mathbf{a} \in Act(Ag, s)$ such that $\mathbf{a}[i] = \alpha[i]$ for each $i \in C$ and $\mathbf{a}[j] = \beta[j]$ for each $j \in \overline{C}$. We denote by $Act(C, S)$ the set $\bigcup_{s \in S} Act(C, s)$. A **path** ρ is an infinite alternated sequence $s_1, \mathbf{a}_1, s_2, \ldots$ of states and tuples in ACT such that for all $i \geq 1$, $t(s_i, \mathbf{a}_i) = s_{i+1}$. If ρ is a path, we denote by ρ^S the sub-sequence of ρ only containing states. If $h \in S^+$ is a finite sequence of states, we say that h is a **history** iff there is a path ρ such that $h = \rho^S_{\leq i}$ for some $i \in \mathbb{N}$. We use H to denote the set of all histories.

2.1. Resource Bounded ATL

In many multi-agent systems, agents are resource-bounded, in the sense that they require resources in order to act. To formalise such notion, in [6] the authors introduced Resource Bounded ATL (RB-ATL for short). RB-ATL is a variant of ATL in which strategic formulae are decorated with bound, i.e., natural numbers vectors of finite size. The intended meaning of a formula $\langle\!\langle C^{\mathbf{b}} \rangle\!\rangle\, \psi$ of RB-ATL can be expressed as *the coalition of agents C has a strategy to achieve the objective ψ whose cost does not surpass* $\mathbf{b}$. We now recall the syntax and semantics of RB-ATL.

Definition 2. *Formulae of RB-ATL are defined by the following grammar:*

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle C^{\mathbf{b}} \rangle\!\rangle \, \mathsf{X}\, \varphi \mid \langle\!\langle C^{\mathbf{b}} \rangle\!\rangle \, \mathsf{G}\, \varphi \mid \langle\!\langle C^{\mathbf{b}} \rangle\!\rangle \, \varphi \, \mathsf{U}\, \varphi$$

where $p \in Ap$, $C \subseteq Ag$, $\mathbf{b}$ is any bound, $\langle\!\langle C^{\mathbf{b}} \rangle\!\rangle$ is the strategic operator, X is the next operator, G is the globally operator, and U is the until operator. We can derive the boolean connectives $\bot$, $\vee$, and $\to$ as usual. We define $\langle\!\langle C^{\mathbf{b}} \rangle\!\rangle\, \mathsf{F}\, \varphi$ as $\langle\!\langle C^{\mathbf{b}} \rangle\!\rangle\, \top\, \mathsf{U}\, \varphi$. We will use φ, ψ, θ, etc., to denote arbitrary formulae.

Formulae of RB-ATL are interpreted over RB-CGSs. These are CGSs in which a cost (a natural number vector) is associated to any agent action at any state. The formal definition follows.

Definition 3. *A Resource Bounded CGS (RB-CGS for short) is a triple $\mathfrak{M} = \langle \mathfrak{G}, r, \mathscr{C} \rangle$ where:*

- *$\mathfrak{G}$ is a CGS as in Definition 1;*

- *$r \geq 1$ is a natural number (the number of resources types);*

- *$\mathscr{C} : S \times act \to \mathbb{N}^r$ is a function associating to any state s and action a, a cost $\mathscr{C}(s, a)$, that is a vector in $\mathbb{N}^r$.*

As in [6] we impose that any agent at any state has at its disposal the idle action $\star$ and that the cost of such action is always $\mathbf{0}$.

Strategies are defined in the standard way as follows.

Definition 4. *A strategy for a coalition of agents C is a function $\sigma_C : H \to Act(C, S)$ mapping a history h to a joint action $\alpha \in Act(C, last(h))$.*

A path $\rho = s_1, \mathbf{a_1}, s_2 \ldots$ is **compatible** with a joint strategy σ_C if for every $i \geq 1$ and every $k \in C$ it holds that $\sigma_C(\rho^S_{\leq i})[k] = \mathbf{a}_i[k]$. We denote with $out(s, \sigma_C)$ the set of all σ_C-compatible paths whose first element is s.

Let s be a state and $\alpha \in Act(C, s)$. The cost of α at s is given by:

$$cost(s, \alpha) = \sum_{i \in C} \mathscr{C}(s, \alpha[i])$$

Let σ_C be a strategy for the coalition C, $\rho = s_1, \mathbf{a}_1, s_2, \ldots$ be a path in $out(s, \sigma_C)$, and $\mathbf{b} \in \mathbb{N}^r$. We say that ρ is $\mathbf{b}$-consistent when for each natural number $n \geq 1$, we have that:

$$\sum_{k=1}^{n} cost(s_k, \sigma_C(\rho^S_{\leq k})) \leq \mathbf{b}$$

A strategy σ_C for a coalition C is $\mathbf{b}$-consistent whenever, for every state s, given any $\rho \in out(s, \sigma_C)$, ρ is $\mathbf{b}$-consistent.

We now define the semantic interpretation of RB-ATL formulae.

Definition 5. *Given a RB-CGS $\mathfrak{M}$, a state s of $\mathfrak{M}$, and a formula φ, the satisfaction relation $\mathfrak{M}, s \models \varphi$ is inductively defined on the structure of φ as follows:*

- *$\mathfrak{M}, s \models \top$ always;*

- *$\mathfrak{M}, s \models p$ iff $p \in L(s)$;*

- *$\mathfrak{M}, s \models \neg\psi$ iff it is not the case that $\mathfrak{M}, s \models \psi$ (denoted $\mathfrak{M}, s \not\models \psi$);*

- *$\mathfrak{M}, s \models \psi \wedge \theta$ iff $\mathfrak{M}, s \models \psi$ and $\mathfrak{M}, s \models \theta$;*

- *$\mathfrak{M}, s \models \langle\langle C^{\mathbf{b}} \rangle\rangle \, \mathrm{X} \, \psi$ iff there is a $\mathbf{b}$-consistent strategy σ_C for the coalition C such that for every path $\rho \in out(s, \sigma_C)$ we have that $\mathfrak{M}, \rho^S_2 \models \psi$;*

- *$\mathfrak{M}, s \models \langle\langle C^{\mathbf{b}} \rangle\rangle \, \mathrm{G} \, \psi$ iff there is a $\mathbf{b}$-consistent strategy σ_C for the coalition C such that for every path $\rho \in out(s, \sigma_C)$ and for every $i \geq 1$, we have that $\mathfrak{M}, \rho^S_i \models \psi$;*

- $\mathfrak{M}, s \models \langle\!\langle C^{\mathbf{b}} \rangle\!\rangle\, \psi \, \mathsf{U}\, \theta$ *iff there is a* $\mathbf{b}$*-consistent strategy* σ_C *for the coalition* C *such that for every path* $\rho \in out(s, \sigma_C)$ *there exists a* $j \geq 1$ *such that* $\mathfrak{M}, \rho_j^S \models \theta$ *and for every* $1 \leq i < j$ *we have that* $\mathfrak{M}, \rho_i^S \models \psi$*.*

We write $\mathfrak{M} \models \varphi$ *and we say that* $\mathfrak{M}$ *satisfies* φ *iff* $\mathfrak{M}, s_I \models \varphi$*.*

In [6] the authors claim that the model-checking problem for RB-ATL can be solved in time that is linear on the size of the model and exponential in the number r of resource bounds. We report the exact statement of the Theorem.

Theorem 1 ([6]). *Given a finite RB-CGS* $\mathfrak{M}$ *and a formula* φ*, there is an algorithm that computes* $[\![\varphi]\!]^{\mathfrak{M}}$ *which runs in* $\mathbf{O}(|\mathfrak{M}|, |\varphi|^{2r+1})$ *where* $|\mathfrak{M}|$ *is the size of the RB-CGS and* r *is its the number of resources.*

3. Case study

The Curiosity rover stands as one of the most sophisticated systems ever deployed for planetary exploration missions. Its primary goals are to capture image data and gather soil and rock samples. Unlike the original model [7], in this scenario, the rover is equipped with autonomous decision-making capabilities, as described in [8]. However, contrary to the scenario in [8], we assume the rover operates with perfect information but factor in the resources necessary to accomplish the mission. We simulate an inspection mission where the Curiosity rover patrols a topological map of the Martian surface.

In Figure 1, we illustrate the model $\mathfrak{M}$ that describes a sample mission for the rover. The mission begins with the rover in state s_I, where it must execute a setup action (*chk*) that involves checking one of the rover's three main components: the arm (*ca*), the mast (*cm*), or the wheels (*cw*). To conserve time and energy, the mechanic (the entity responsible for performing the setup checks and making any necessary adjustments) conducts only one setup operation per mission. Specifically, checking the arm requires 5 units of time and 3 units of energy, inspecting the mast takes 2 units of time and 4 units of energy, and examining the wheels consumes 3 units of time and 5 units of energy. These varying requirements are due to the differing complexities of each action. The rover must validate the setup operation. If, in the initial state, the mechanic chooses to check the arm, denoted as *ca*, then in the subsequent state, s_1, the rover needs to perform the *ca* action to proceed from s_1. The same logic applies to the other two setup actions. After the selection, the mechanic can either choose to check and potentially correct the component (action *ok*) or decline the operation (action *nok*). In the former case, the rover can continue the mission, whereas in the latter, the mission ends with an error. If the mechanic collaborates, the rover can begin the mission. We denote the state where the rover is at the base camp as s_4. The objective is for the rover to move from its initial position, take a picture of a sample rock on Mars, and then return to the starting position. Specifically, from state s_4, the rover can decide to move left (L) to state s_6 or right (R) to state s_7. In either of these states, the rover can take a photo of a sample rock (action mp). Following this step, the rover must conclude the mission by returning to the base camp. To do this, it needs to perform the complementary move action to return to the previously visited state (R from s_6 and L from s_7).

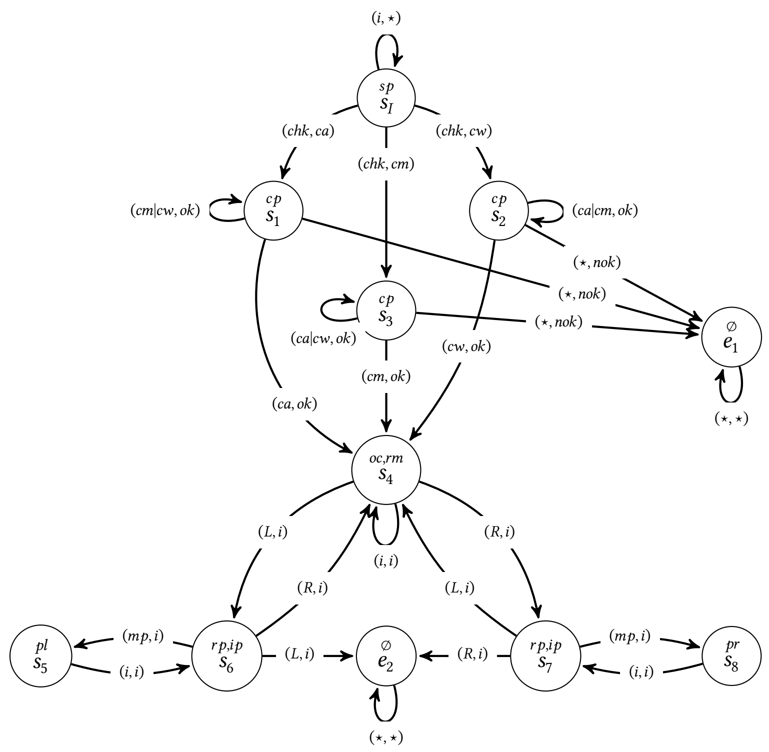


Figure 1: The rover's mission where i stands for idle and $\star$ for any action. The atoms have the following acronyms: sp as starting position, cp as check phase, oc as ok check phase, rm as ready to mission, rp as ready to make a picture, ip as in position, pl as picture left, and pr as picture right.

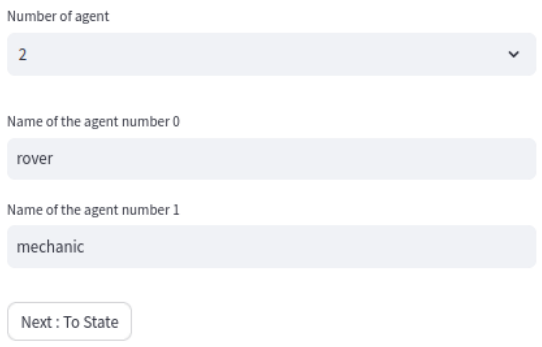
Given the model M, we can define several specifications. For example, the ATL specification that describes the rover mission is

$$\varphi_1 = \langle\!\langle rover \rangle\!\rangle\, F((oc \wedge rm) \wedge \langle\!\langle rover \rangle\!\rangle\, F((pl \vee pr) \wedge \langle\!\langle rover \rangle\!\rangle\, F(oc \wedge rm)))$$

In words, this formula means that there exists a strategy for the rover such that it will eventually be ready to start the mission, can take a picture of a sample rock, and can return to the base camp. In the previous formula, we assume that the mechanic may choose not to cooperate. In this scenario, it becomes impossible for the rover to complete the mission. Thus, the formula φ_1 is false in the model $\mathfrak{M}$. However, if we assume the mechanic cooperates (i.e., the mechanic checks a component and corrects it if necessary via the action *ok*), we can rewrite the specification as an RB-ATL formula:

$$\varphi_2 = \langle\!\langle rover, mechanic^{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle} \rangle\!\rangle\, F((oc \wedge rm) \wedge \langle\!\langle rover \rangle\!\rangle\, F((pl \vee pr) \wedge \langle\!\langle rover \rangle\!\rangle\, F(oc \wedge rm)))$$

where $\mathbf{b}_1$ sets the resource bound for time, while $\mathbf{b}_2$ sets the resource bound for energy. If $\mathbf{b}_1$ and $\mathbf{b}_2$ are sufficient for the mechanic to select the actions necessary to achieve its strategic objectives, then the rover has a strategy to satisfy the formula by relying on the mechanic's cooperation, that is $\mathfrak{M}$ satisfies φ_2. In fact, a simple strategy σ can be defined as follows: $\sigma(s_I) = chk$,


Number of agent

2

Name of the agent number 0

rover

Name of the agent number 1

mechanic

Figure 2: VITAMIN's GUI: Step (i) - Number and Names of Agents.

$\sigma(s_I s_j) = ca$, $\sigma(s_I s_j s_k) = cm$, $\sigma(s_I s_j s_k s_r) = cw$, $\sigma(s_I s_j s_4) = L$, $\sigma(s_I s_j s_4 s_6) = mp$, $\sigma(s_I s_j s_4 s_6 s_5) = i$, and $\sigma(s_I s_j s_4 s_6 s_5 s_6) = R$, where $j, k, r \in \{1, 2, 3\}$.

4. VITAMIN in action

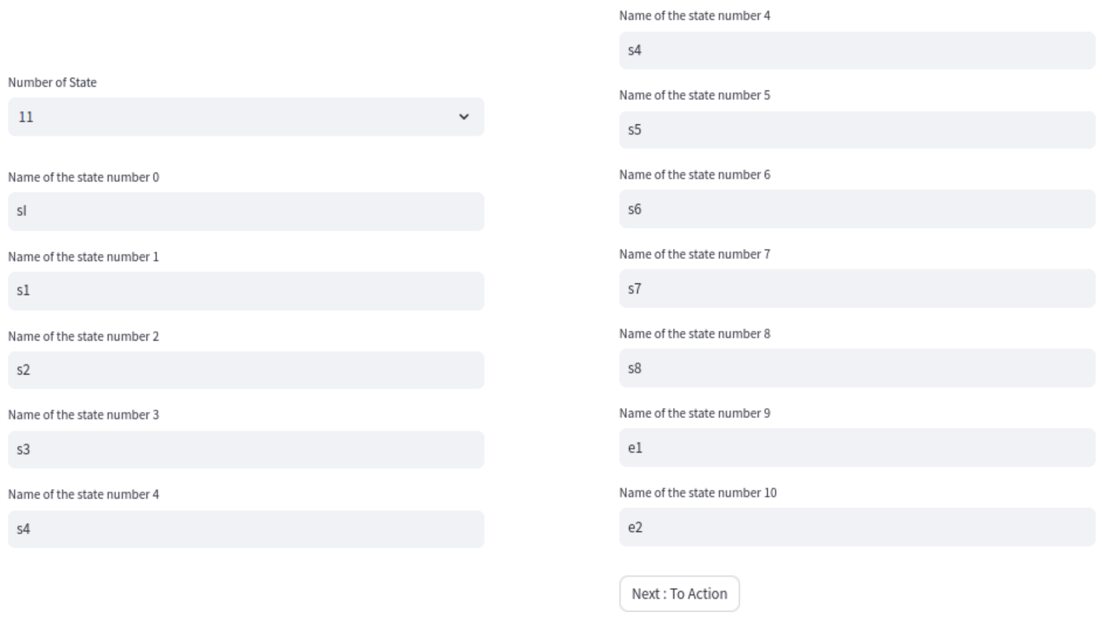
In this section, we demonstrate how VITAMIN can be used to verify the previously presented case study. We do this by following all the steps required to build the CGS shown in Figure 1. Note that, in VITAMIN, there are two possible ways to create CGSs. The first method is for expert users and requires the user to upload a properly formatted CGS according to VITAMIN's input format. The second method is for non-expert users and allows VITAMIN to guide the user through the CGS creation process in a step-by-step fashion.

Agents. The first aspect VITAMIN queries the user about is the number of agents in the MAS. Figure 2 reports a screenshot of VITAMIN's GUI. In this step, the user can decide how many agents are in the MAS and which are their names. Considering the rover case study, in this step, we report the presence of two agents, called *rover* and *mechanic*, respectively.

States. Once the agents have been decided, VITAMIN asks for the states of the CGS. Figure 3 reports this step. In our scenario, according to Figure 1, we have 11 different states, that is the initial state s_I, the normal states s_1 to s_8, and the error states e_1 and e_2.

Atoms. States may contain atomic propositions, thus, after gathering the states, VITAMIN expects the user to provide the atoms holding in such states. Figure 4 reports this step in VITAMIN's GUI. Considering once more the rover case study, in this step we may find 8 different atomic propositions.

Labelling. Naturally, atomic propositions make sense when linked to the states of our model. Thus, after gathering the atoms, VITAMIN requires the user to report which atoms hold in which states of the CGS. Figure 5 report this step for the rover case study. Here, we can observe how



Number of State: 11

Name of the state number 0: sl

Name of the state number 1: s1

Name of the state number 2: s2

Name of the state number 3: s3

Name of the state number 4: s4

Name of the state number 4: s4

Name of the state number 5: s5

Name of the state number 6: s6

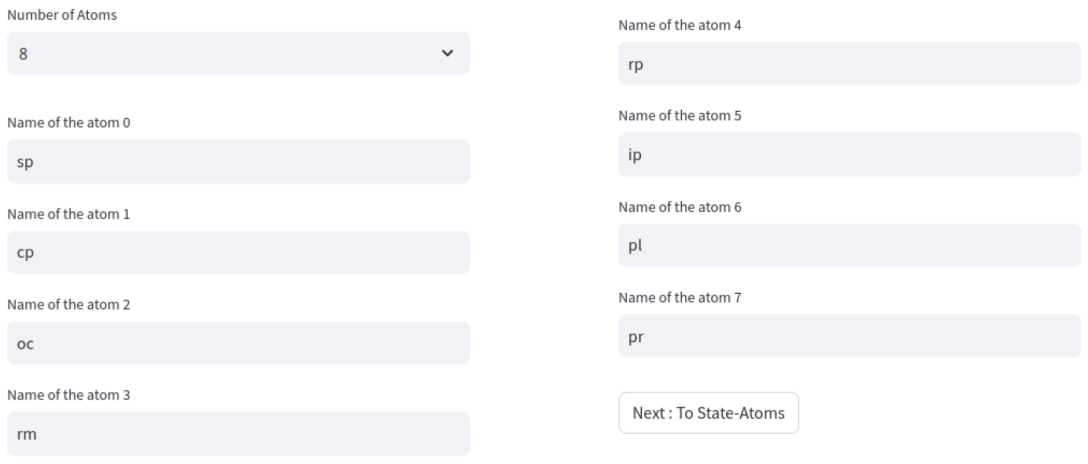
Name of the state number 7: s7

Name of the state number 8: s8

Name of the state number 9: e1

Name of the state number 10: e2

Figure 3: `VITAMIN`'s GUI: Step (ii) - Number and Names of States.



Number of Atoms: 8

Name of the atom 0: sp

Name of the atom 1: cp

Name of the atom 2: oc

Name of the atom 3: rm

Name of the atom 4: rp

Name of the atom 5: ip

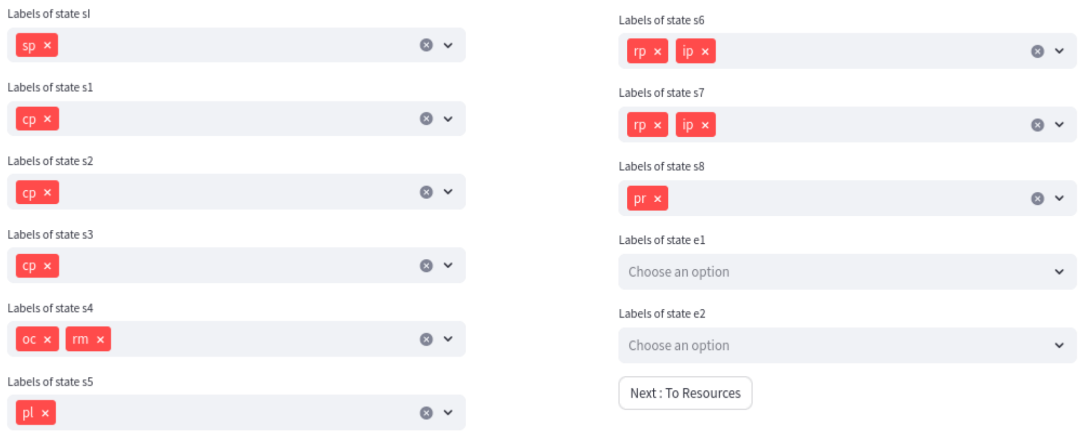
Name of the atom 6: pl

Name of the atom 7: pr

Figure 4: `VITAMIN`'s GUI: Step (iii) - Number and Names of Atoms.

the states are labelled according to Figure 1, where in each state the user can list all the atoms holding in such a state (e.g., in state s_4 the *oc* and *rm* atoms hold).

Resources. Since we are talking about CGSs with resources (*i.e.*, RB-CGSs), after populating the model with states and atoms, the user needs to insert the number and names of the resources to be considered in the RB-CGS. Figure 6 reports the corresponding step in `VITAMIN`'s interface, where *time* and *energy* are listed as the two resources to be considered in the model.


Labels of state sI: sp

Labels of state s1: cp

Labels of state s2: cp

Labels of state s3: cp

Labels of state s4: oc, rm

Labels of state s5: pl

Labels of state s6: rp, ip

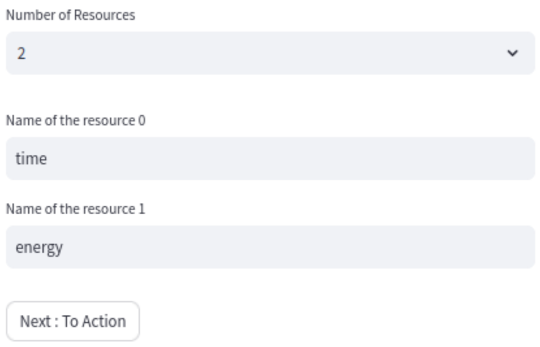
Labels of state s7: rp, ip

Labels of state s8: pr

Labels of state e1: Choose an option

Labels of state e2: Choose an option

Figure 5: VITAMIN's GUI: Step (iv) - Labelling.


Number of Resources: 2

Name of the resource 0: time

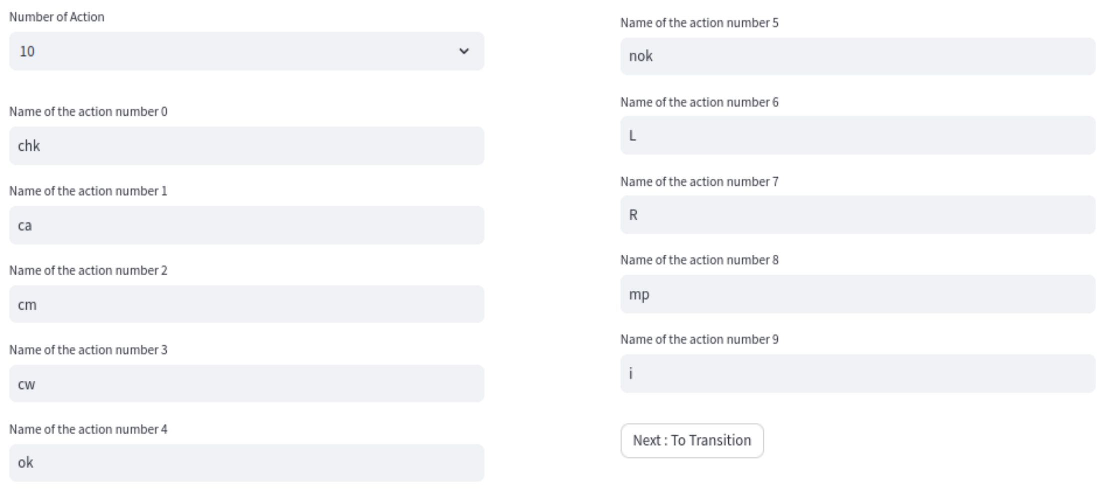
Name of the resource 1: energy

Figure 6: VITAMIN's GUI: Step (v) - Number and Names of Resources.

Actions. After the agents, states, atoms, and resources have been gathered, VITAMIN continues by asking for the actions to be performed by the agents. Figure 7 reports the list of actions for the rover case study, where we have 10 different actions to be performed by the *rover* and *mechanic* agents.

Transitions. Once the actions are given, VITAMIN expects the user to insert the transitions amongst the different states of the model. Figure 8 and Figure 9 report a snippet of the transitions of the rover and mechanic agents, respectively. For lack of space, only a subset of the transitions is reported. Note that, thanks to VITAMIN's GUI, the user can easily and intuitively insert the actions each agent can perform in a state to reach another state. For instance, in Figure 8, the user can insert that the rover agent can perform the action *chk* in state s_I to reach states s_1, s_2, and s_3 (as reported in Figure 1).

Costs. Last, but not least, to complete all information on the model, VITAMIN asks the user to report the costs of the transitions. Note that, the cost is determined by the actions and the



Number of Action

10

Name of the action number 0

chk

Name of the action number 1

ca

Name of the action number 2

cm

Name of the action number 3

cw

Name of the action number 4

ok

Name of the action number 5

nok

Name of the action number 6

L

Name of the action number 7

R

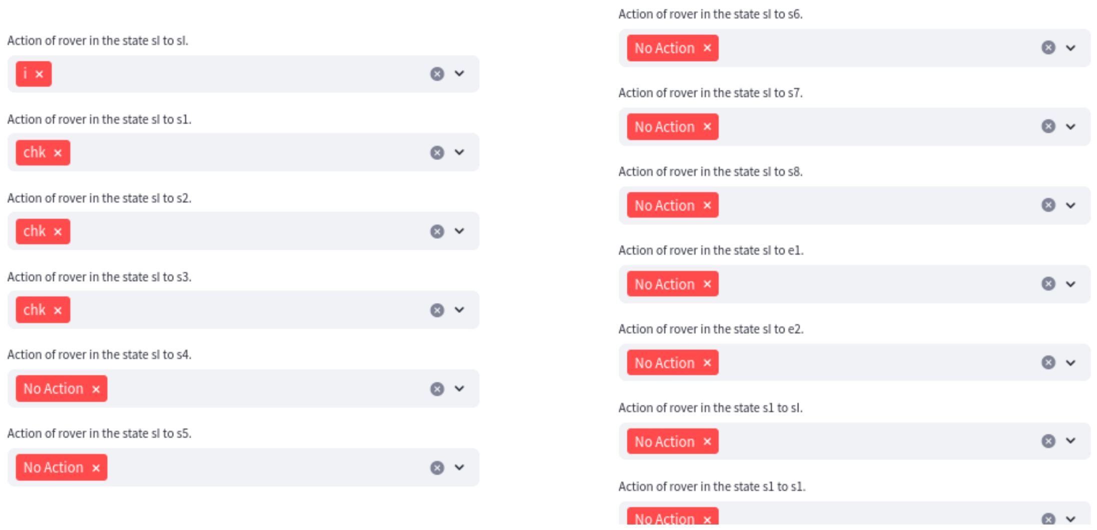
Name of the action number 8

mp

Name of the action number 9

i

Figure 7: VITAMIN's GUI: Step (vi) - Actions.



Action of rover in the state sI to sI.

i ×

Action of rover in the state sI to s1.

chk ×

Action of rover in the state sI to s2.

chk ×

Action of rover in the state sI to s3.

chk ×

Action of rover in the state sI to s4.

No Action ×

Action of rover in the state sI to s5.

No Action ×

Action of rover in the state sI to s6.

No Action ×

Action of rover in the state sI to s7.

No Action ×

Action of rover in the state sI to s8.

No Action ×

Action of rover in the state sI to e1.

No Action ×

Action of rover in the state sI to e2.

No Action ×

Action of rover in the state s1 to sI.

No Action ×

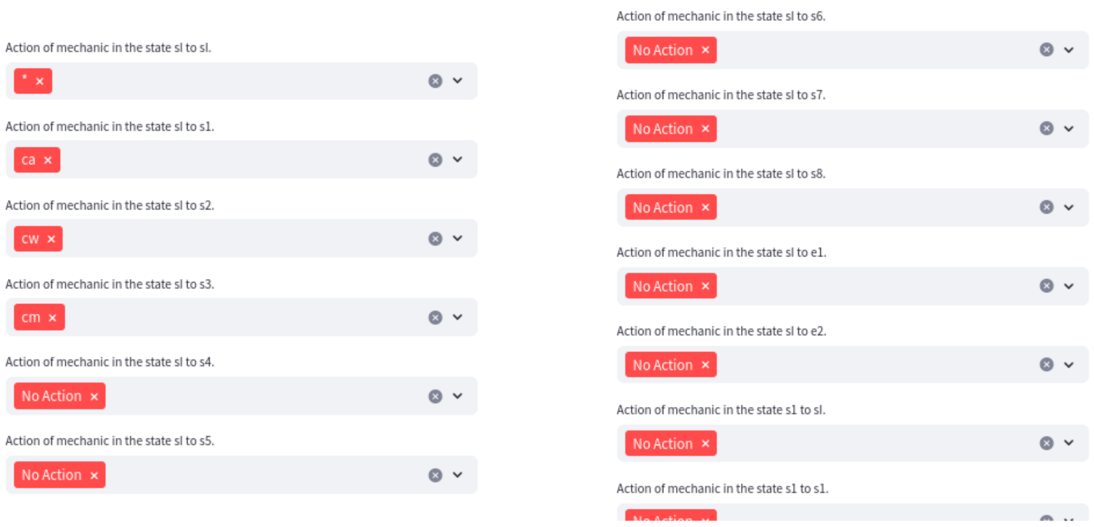
Action of rover in the state s1 to s1.

No Action ×

Figure 8: VITAMIN's GUI: Step (vii) - Transitions of the rover agent.

states such actions are performed into. Figure 10 reports a snippet of the costs introduced in the context of the rover case study. Here, we can see how the costs matches the ones presented in Section 3, when we presented the notion of time and energy.

Visualisation. Before concluding with the verification of the properties of interest, VITAMIN allows the user to validate the model created by following the previous steps. In particular, VITAMIN shows the resulting model via its GUI to the user.



Action of mechanic in the state sI to sI.

* ×

Action of mechanic in the state sI to s1.

ca ×

Action of mechanic in the state sI to s2.

cw ×

Action of mechanic in the state sI to s3.

cm ×

Action of mechanic in the state sI to s4.

No Action ×

Action of mechanic in the state sI to s5.

No Action ×

Action of mechanic in the state sI to s6.

No Action ×

Action of mechanic in the state sI to s7.

No Action ×

Action of mechanic in the state sI to s8.

No Action ×

Action of mechanic in the state sI to e1.

No Action ×

Action of mechanic in the state sI to e2.

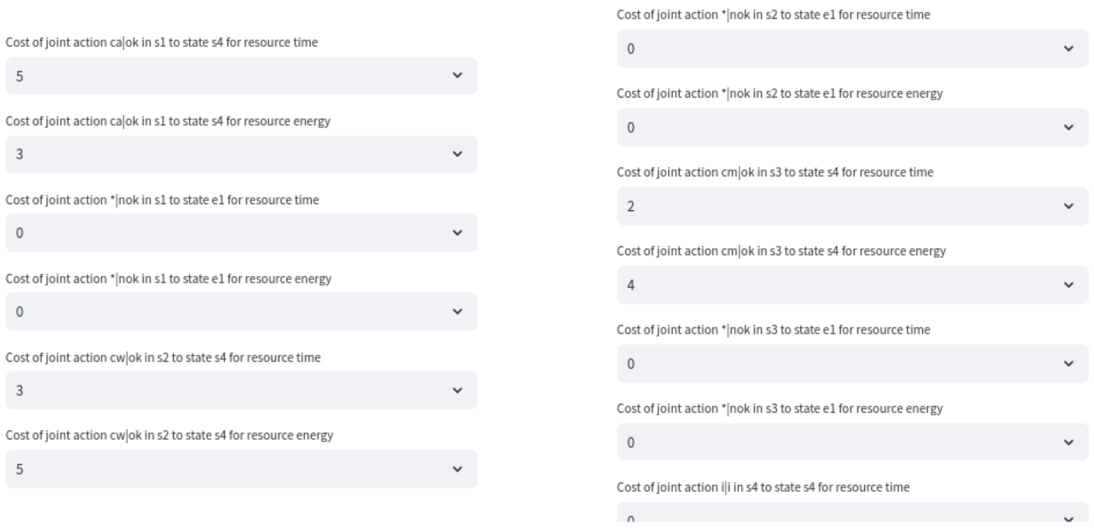
No Action ×

Action of mechanic in the state s1 to sI.

No Action ×

Action of mechanic in the state s1 to s1.

Figure 9: VITAMIN's GUI: Step (vii) - Transitions of the mechanic agent.



Cost of joint action ca|ok in s1 to state s4 for resource time

5

Cost of joint action ca|ok in s1 to state s4 for resource energy

3

Cost of joint action *|nok in s1 to state e1 for resource time

0

Cost of joint action *|nok in s1 to state e1 for resource energy

0

Cost of joint action cw|ok in s2 to state s4 for resource time

3

Cost of joint action cw|ok in s2 to state s4 for resource energy

5

Cost of joint action *|nok in s2 to state e1 for resource time

0

Cost of joint action *|nok in s2 to state e1 for resource energy

0

Cost of joint action cm|ok in s3 to state s4 for resource time

2

Cost of joint action cm|ok in s3 to state s4 for resource energy

4

Cost of joint action *|nok in s3 to state e1 for resource time

0

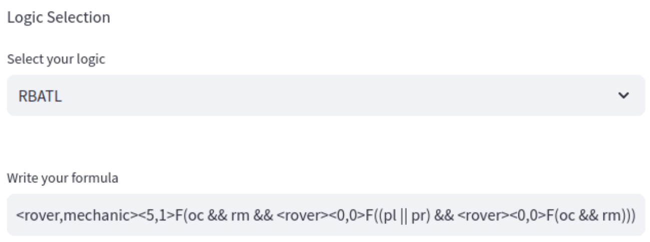
Cost of joint action *|nok in s3 to state e1 for resource energy

0

Cost of joint action i|i in s4 to state s4 for resource time

Figure 10: VITAMIN's GUI: Step (viii) - Costs of the transitions.

Formula and Verification. Finally, after the model is created and validated by the user, the guided process concludes by asking the user to fill in the formula to verify on the model. In this step, as reported in Figure 11, the user can select the logic to use (in this case RBATL), and insert the formula to verify on the model; in this case, as an example, we reported φ_2 with $\mathbf{b}_1 = 5$ and $\mathbf{b}_2 = 1$. After filling these two fields, VITAMIN runs the corresponding model checking algorithm for the logic and model selected, and returns back to the user the result of the verification.


Logic Selection

Select your logic

RBATL

Write your formula

```
<rover,mechanic><5,1>F(oc && rm && <rover><0,0>F((pl || pr) && <rover><0,0>F(oc && rm)))
```

Figure 11: VITAMIN's GUI: Step (ix) - Logic and Verification.

4.1. Experimental results

Other than showing how VITAMIN can support a user in the definition of a model to verify, we also want to report the experimental results obtained on the verification of the actual rover case study. We tested our tool over the rover's mission, on a machine with the following specifications: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 4 cores 8 threads, 16 GB RAM DDR4. Specifically, we verified formula φ_1 and φ_2 and reported the results so obtained in Table 1.

Formula	Bounds	Result	Time (sec)
φ_1	-	$\perp$	0.003
φ_2	$\mathbf{b}_1 = 5, \mathbf{b}_2 = 1$	$\top$	0.082
φ_2	$\mathbf{b}_1 = 2, \mathbf{b}_2 = 2$	$\perp$	0.024

Table 1
Results of the verification process for different formulas and bounds

As we can see, the verification can be concluded within 0.1 seconds. In the case of φ_1, this requires less than 4 milliseconds. For φ_2, the verification time ranges from 0.024 to 0.082 seconds, depending on the values associated with $\mathbf{b}_1$ and $\mathbf{b}_2$. Now, we can discuss the outcome of the tool. As discussed in Section 3, φ_1 should be false, and in fact our tool returns false. The formula φ_2 with enough resources for the mechanic (like $\mathbf{b}_1 = 5$ and $\mathbf{b}_2 = 1$) is verified in $\mathfrak{M}$, as it is confirmed by our tool; while with less resources (like $\mathbf{b}_1 = 2$ and $\mathbf{b}_2 = 2$) φ_2 is not verified in $\mathfrak{M}$, also confirmed by our tool. These experiments, although limited to the case study presented in this paper, demonstrate the applicability and feasibility of VITAMIN in tackling resource-bound verification of MAS.

5. Conclusions and Future Work

In this paper, we presented how VITAMIN can guide the verification of an interesting case study based on the Curiosity rover. We demonstrated how VITAMIN supports the user in all specification phases, including the construction, population, and verification of the CGS against a formal specification. We conducted experiments with VITAMIN and reported the results, confirming its applicability. Furthermore, we evaluated not only the usability of the tool but also its performance in the formal verification of the case study.

As a future direction, we plan to further test VITAMIN on more complex case studies and to gather feedback from the MAS community on its usability. This feedback will be used to further improve and extend VITAMIN.

References

[1] C. Baier, J.-P. Katoen, Principles of model checking, MIT press, 2008.

[2] A. Lomuscio, H. Qu, F. Raimondi, MCMAS: an open-source model checker for the verification of multi-agent systems, Int. J. Softw. Tools Technol. Transf. 19 (2017) 9–30. URL: https://doi.org/10.1007/s10009-015-0378-x. doi:10.1007/S10009-015-0378-X.

[3] D. Kurpiewski, W. Jamroga, M. Knapik, STV: model checking for strategies under imperfect information, in: E. Elkind, M. Veloso, N. Agmon, M. E. Taylor (Eds.), Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 2372–2374. URL: http://dl.acm.org/citation.cfm?id=3332116.

[4] A. Ferrando, V. Malvone, VITAMIN: A compositional framework for model checking of multi-agent systems, CoRR abs/2403.02170 (2024). URL: https://doi.org/10.48550/arXiv.2403.02170. doi:10.48550/ARXIV.2403.02170. arXiv:2403.02170.

[5] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (2002) 672–713. URL: https://doi.org/10.1145/585265.585270. doi:10.1145/585265.585270.

[6] N. Alechina, B. Logan, N. H. Nga, A. Rakib, Resource-bounded alternating-time temporal logic, in: W. van der Hoek, G. A. Kaminka, Y. Lespérance, M. Luck, S. Sen (Eds.), 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3, IFAAMAS, 2010, pp. 481–488. URL: https://dl.acm.org/citation.cfm?id=1838274.

[7] NASA, Mars curiosity rover, 2012. URL: https://mars.nasa.gov/msl/home/.

[8] A. Ferrando, V. Malvone, Towards the verification of strategic properties in multi-agent systems with imperfect information, in: N. Agmon, B. An, A. Ricci, W. Yeoh (Eds.), Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023, ACM, 2023, pp. 793–801. URL: https://dl.acm.org/doi/10.5555/3545946.3598713. doi:10.5555/3545946.3598713.