# Group Synthesis for Alternating-Time Temporal Logic

A. V. Jones[1⋆], M. Knapik[2], A. Lomuscio[3], and W. Penczek[2,4]

[1] Thales, Manor Royal, Crawley, UK
andrewj@fmethods.com
[2] Institute of Computer Science, PAS, Warsaw, Poland
{mknapik,penczek}@ipipan.waw.pl
[3] Department of Computing, Imperial College London, UK
A.Lomuscio@imperial.ac.uk
[4] University of Natural Sciences and Humanities, ICS, Siedlce, Poland

**Abstract** We present an extension of Alternating-time Temporal Logic ATL, called ATLP (Parametric ATL), where parameters are allowed in place of concrete groups of agents. We devise a procedure to find all instantiations for the parameters in a given formula $\phi$ of ATLP so that $\phi$ is true in a given model. We propose a formalisation of the problem and symbolic algorithms for its solution. We discuss an experimental implementation of the approach on top of the open-source model checker MCMAS and demonstrate the benefits of the technique through experimental results.

## 1 Introduction

Multi-agent systems (MAS) consist of multiple entities called *agents*, often assumed to be intelligent, and able to interact with each other and with the environment. More and more practical problems are being modeled and solved under paradigms related to multi-agent systems. The examples of their industrial applications include space mission planning and air control [6,19,20], defense and security projects [8,18], logistics and production planning [9,10] and many others. According to the 2008 review [17], companies like Rockwell Automation, Volkswagen, SkodaAuto, Daimler AG, Magenta, NASA and JPL either routinely use or are actively developing multi-agent based solutions.

Given the proliferation of MAS in the modern world, the need of their automated verification, i.e., ensuring of the compliance with their specification, becomes an important task. Model checking is a technique in which the system in question is described by using a formal model and its specification is expressed by a formula of modal logic. More formally, for a model $\mathcal{M}$, its state $g$ and a property $\phi$ we wish to automatically check whether $\phi$ holds in $g$, denoted by $\mathcal{M}, g \models \phi$. In this paper we are interested in group synthesis for MAS.

⋆ This work was conducted while the first author was a Ph.D. student in the Department of Computing at Imperial College London.

### 1.1 Rationale for Group Synthesis

While the approach of model checking has proven to be very successful, it suffers from some limitations. Consider a game in which two arbitrarily chosen groups compete to capture a given position, and consider Alternating-time Temporal Logic as the intended specification language. Naturally, the members of each team are allowed to cooperate with their teammates. The goal of each team is to reach the target location, stay there, and prevent the opponent from entering. If we have the prior knowledge that $Team_1$ consists of the agents numbered with $1, 3, 5$ and $Team_2$ consists of the remaining agents $2, 4, 6$, then the situation can be modeled using an appropriate model $\mathcal{M}$ and the following ATL formula:

$$\phi = \langle\langle Team_1 \rangle\rangle F(\langle\langle Team_1 \rangle\rangle G\ target \wedge \neg\langle\langle Team_2 \rangle\rangle F\ target),$$

This can be automatically verified using a model checker such as MOCHA [2] or MCMAS [13]. The question we pose here, however, is *who can participate in* $Team_1$ *and* $Team_2$ *so that the* $\phi$ *property is true?* To deal with this, we allow for the presence of free variables (called *parameters*) in the formulae of ATL. In this way we are able to consider the formulae of type:

$$\psi = \langle\langle Y_1 \rangle\rangle F(\langle\langle Y_1 \rangle\rangle G\ target \wedge \neg\langle\langle Y_2 \rangle\rangle F\ target),$$

where $Y_1, Y_2$ are parameters.

Let $\upsilon$ denote an assignment of the parameters with groups of agents, and denote by $\mathcal{M}, g \models_\upsilon \psi$ that $\psi$ holds in the state $g$ of model $\mathcal{M}$ after the substitution of the parameters consistent with $\upsilon$. Our aim is to characterise *all* the substitutions under which $\psi$ is true in $\mathcal{M}$. To this end, we propose a formalism in which $f_\psi$ is defined as a characteristic function assigning to each state $g$ the set of all assignments under which $\psi$ becomes true in $g$, i.e., $\upsilon \in f_\psi(g) \iff \mathcal{M}, g \models_\upsilon \psi$.

To illustrate the concept, consider the case of the $\psi$ formula above. Our main concern is how to synthesise the $f_\psi$ function efficiently. A brute-force approach would consist in substituting $Y_1$ and $Y_2$ with all the possible pairs of subsets of the set of Agents, which requires $2^{2|\text{Agents}|}$ separate calls to model checking routines for non-parametric formulas. To alleviate this, we introduce symbolic algorithms operating on functions instead of sets of states. Let

$$\psi_{inn} = \langle\langle Y_1 \rangle\rangle G\ target \wedge \neg\langle\langle Y_2 \rangle\rangle F\ target$$

denote the inner formula of $\psi$. Firstly, we can compute by means of the brute-force approach the $f_{\psi_{inn}}$ function associated with the inner formula. Then, we build the characteristic function $f_\psi$ by computing $f_{\langle\langle \Gamma \rangle\rangle F \psi_{inn}}$ for each $\Gamma \subseteq$ Agents. We define symbolic parametric algorithms for calculating the $f_{\langle\langle \Gamma \rangle\rangle F \psi_{inn}}$ function in a single step, therefore in order to build $f_\psi$ we need the total of $2^{|\text{Agents}|+1}$ calls (the inner formula) to the non-parametric and $2^{|\text{Agents}|}$ calls to parametric routines. Combined with Reduced Ordered Binary Decision Diagrams (ROBDDs) used to represent models in a compact way, this method allows for an automatic synthesis of groups for properties with several group parameters.

### 1.2 Related Work and Paper Outline

The problem of synthesis of the set of group assignments under which a given formula becomes true in a selected model was first introduced in [3] in the context of a parametric extension of LTL. A similar question concerning parametric version of (resource-bounded) ATL formulae was posed in [15], but the authors proposed a brute-force solution only. In [5] a related question, namely the size of a minimal coalition able to reach a winning objective, was investigated. The results presented in this paper are related to our earlier approach [11], where a parameter synthesis was explored in the context of a temporal-epistemic logic. We are not aware of other approaches to this problem different from the brute-force one.

The rest of the paper is organised as follows. In the next section we present ATLP, a parametric extension of ATL defined on Interpreted Systems. Section 3 formally defines the task of group synthesis in the setting of ATLP and presents our approach to solving this problem in a form of symbolic, fixed-point algorithms. An experimental evaluation of proposed approach is presented in Section 4. The paper concludes with a brief summary and future work directions.

## 2 Interpreted Systems and ATLP Logic

In this section we introduce Parametric Alternating-time Temporal Logic (ATLP). We give semantics of ATLP over models based on Interpreted Systems. Most of the formal definitions are borrowed directly from [14], with some natural extensions to deal with parameters and their assignments.

### 2.1 Interpreted Systems

Interpreted systems (IS, for short) are an established and convenient formalism for specifying multi-agent systems and for dealing with their strategic, temporal, and epistemic properties. Below, we define all the components of IS.

Let Agents $= \{1, \ldots, k\}$, where $k \in \mathbb{N}$, be a set of *agents*, and $Act_i$ be a finite set of the *actions* of agent $i$. Each agent $i \in$ Agents is characterised by a finite set $L_i$ of *local states* and a *protocol function* $P_i : L_i \to 2^{Act_i}$. The protocol function assigns a set of allowed actions to each local state. We also introduce one special agent $E$, which is used to model the environment of the system; its actions, local states, and a protocol function are denoted by $Act_E$, $L_E$, and $P_E$, respectively. By $GS = L_1 \times \ldots \times L_n \times L_E$ we denote the set of *global states* and by $ACT = Act_1 \times \ldots \times Act_k \times Act_E$ we mean the set of *joint actions*. For a given global state $g = (g_1, \ldots, g_k, g_E)$ and agent $i \in$ Agents, let $l_i(g) = g_i$. We define an *evolution function* $t_i : L_i \times L_E \times Act_1 \times \ldots \times Act_k \times Act_E \to L_i$ for each $i \in$ Agents. This function depends on a current local state of the agent and of the environment, and actions selected by all the agents in the system. For a group of agents $\Gamma \subseteq$ Agents let us define a *group protocol function* $P_\Gamma : GS \to \Pi_{i \in \Gamma} 2^{Act_i}$ such that $P_\Gamma(g) = \Pi_{i \in \Gamma} P_i(l_i(g))$ for each global state $g \in GS$. If

$\Gamma = \text{Agents} \cup \{E\}$, then $P_\Gamma$ is denoted simply by $P$ and called the *global protocol function*. Notice that $P(g)$ consists of all the joint actions allowed in $g$.

The function $t$ describing the evolution of the whole system is defined using the $t_i$ functions. Namely, for each global state $g$ and a joint action $a \in P(g)$ we have that $t(g,a) = g'$ iff $t_i(l_i(g), l_E(g), a) = l_i(g')$ for each $g' \in GS$ and $i \in \text{Agents}$. In what follows, we also use the notation $g \xrightarrow{a} g'$ in place of $t(g,a) = g'$. By $I \subseteq L_1 \times \ldots \times L_n \times L_E$ we denote the set of *initial states*. The set of *reachable states* $G$ is defined as the subset of the global states $GS$ that can be obtained via a consecutive application of the $t$ function starting from any initial state in $I$. Moreover, let $\mathcal{PV}$ be a finite set of propositional variables. A *valuation function* $\mathcal{L} : G \to 2^{\mathcal{PV}}$ labels each reachable state with a subset of $\mathcal{PV}$.

Formally, a tuple $\text{IS} = \langle (L_i, Act_i, P_i, t_i)_{i \in \text{Agents} \cup \{E\}}, I, \mathcal{L} \rangle$ is called an *interpreted system*. The labeled transition system $\text{M}_{\text{IS}} = (G, t, \mathcal{L})$ associated with IS is called its *model*.

*Example 1 (Interpreted System).* Consider the two-agent (we have omitted the Environment) model presented in Figure 1. In this case $\mathcal{PV} = \{p\}$ and the labeling is as follows: $\mathcal{L}(w_0) = \emptyset$ and $\mathcal{L}(w_1) = \mathcal{L}(w_2) = \{p\}$. Moreover, $Act_1 = \{a, b\}$, $Act_2 = \{x, y\}$, and the local protocols allow any local move in each state, i.e., $P(w_i) = \{a, b\} \times \{x, y\}$ for each $i \in \{0, 1, 2\}$. The labels on each edge correspond to the joint actions under which the transition can be performed, e.g., we have $w_1 \xrightarrow{ax} w_2$ and $w_1 \xrightarrow{bx} w_2$.

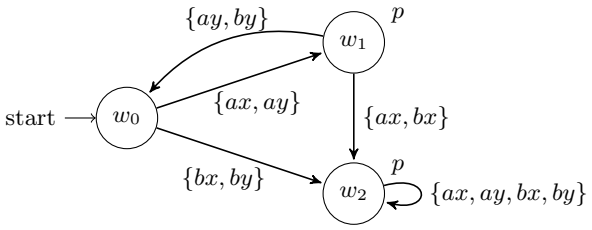


Figure 1: The interpreted system used in Examples 1–4.

By a *run* we mean an infinite sequence of states $\lambda = g_0, g_1, \ldots$ such that $\lambda \in G^\omega$ and for each $i \in \mathbb{N}$ there exists a joint action $a$ satisfying $g_{i+1} \xrightarrow{a} g_i$. Let $\lambda_i = g_i$, $\lambda_0^i = g_0, g_1, \ldots, g_i$ and $\lambda_i^\infty = g_i, g_{i+1}, \ldots$ for any $i \in \mathbb{N}$. A finite prefix of some run is called a *finite run*.

### 2.2 Parametric Alternating-time Temporal Logic

In this section we introduce an extension of Alternating-time Temporal Logic ATL with parameters allowed in the place of concrete groups of agents.

**Definition 1 (ATLP syntax).** *Let* GVars *be a finite set of parameters. The BNF-grammar of Parametric Alternating-time Temporal Logic ATLP is given*

*as follows.*

$$\phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid \langle\langle\chi\rangle\rangle X\phi \mid \langle\langle\chi\rangle\rangle G\phi \mid \langle\langle\chi\rangle\rangle \phi U \phi$$
$$\chi ::= \Gamma \mid Y$$

*where* $p \in \mathcal{PV}$, $\Gamma \subseteq \mathrm{Agents}$, $\Gamma \neq \emptyset$, *and* $Y \in \mathrm{GVars}$.

The set of the ATLP formulae without the parameters reduces to the standard logic ATL. The double brackets serve as a path quantifier, i.e., $\langle\langle\Gamma\rangle\rangle\psi$ is read as "group $\Gamma$ has a strategy to enforce $\psi$". The temporal modalities $X, G, U$ stand for *next*, *globally*, and *until*, respectively.

The formulae of ATLP are interpreted with respect to the assignments of the parameters and strategies for groups of agents.

For an agent $i \in \mathrm{Agents}$ we define a *strategy* as a function $f_i : L_i \to Act_i$ such that $f_i(l_i(g)) \in P_i(l_i(g))$ for each $g \in G$. Intuitively, $f_i$ assigns to each local state of the agent $i$ an action allowed by its protocol. Note that this choice of the definitions of model and strategy is compatible with the concept of imperfect information and memoryless semantics of ATL [14]. Let $\Gamma \subseteq \mathrm{Agents}$ be a non-empty group of agents and $g \in G$. Note that if $a \in P_\Gamma(g)$ and $b \in P_{\mathrm{Agents}\setminus\Gamma}(g)$, then $a$ and $b$ can be uniquely combined into one joint action $c \in P(g)$, denoted by $(a, b)$, such that $c_{|\Gamma} = a$ and $c_{|\mathrm{Agents}\setminus\Gamma} = b$. Let $F_\Gamma = \{f_i \mid i \in \Gamma\}$ be a set of strategies, called a *strategy for* $\Gamma$. For each $g \in G$ we can define a unique joint action $a \in P_\Gamma(g)$ for $\Gamma$ such that $f_i(g)$ for each $i \in \Gamma$ is the $i$–th component of $a$. We denote this action as $F_\Gamma(g)$. The set $out(g, F_\Gamma)$ of the *outcomes* of $F_\Gamma$ from the state $g \in G$ is defined as follows. A run $\lambda$ is in $out(g, F_\Gamma)$ iff $\lambda_0 = g$ and for each $i \in \mathbb{N}$ there is a joint action $b \in P_{\mathrm{Agents}\setminus\Gamma}(g)$ such that $\lambda_i \xrightarrow{(F_\Gamma(g),b)} \lambda_{i+1}$. Intuitively, $out(q, F_\Gamma)$ consists of all the possible runs such that at each step the agents from $\Gamma$ behave according to the set of strategies $F_\Gamma$ while the remaining agents have the full choice of moves.

In the semantics of ATLP we use the notion of an assignment of the parameters (called *group assignment*) $\upsilon : \mathrm{GVars} \to 2^{\mathrm{Agents}} \setminus \{\emptyset\}$. The set of all group assignments is denoted by GroupVals.

**Definition 2 (ATLP semantics).** *Let* $\mathrm{IS} = \langle (L_i, Act_i, P_i, t_i)_{i\in\mathrm{Agents}\cup\{E\}}, I, \mathcal{L}\rangle$ *be an interpreted system,* $\mathrm{M_{IS}} = (G, t, \mathcal{L})$ *be its model,* $g \in G$ *be a state, and* $\upsilon$ *be a group assignment. The relation* $\models_\upsilon$ *is recursively defined as follows:*

- $\mathrm{M_{IS}}, g \models_\upsilon p$ *iff* $p \in \mathcal{L}(g)$ *for* $p \in \mathcal{PV}$,
- $\mathrm{M_{IS}}, g \models_\upsilon \neg\phi$ *iff* $\mathrm{M_{IS}}, g \not\models_\upsilon \phi$,
- $\mathrm{M_{IS}}, g \models_\upsilon \phi \vee \psi$ *iff* $\mathrm{M_{IS}}, g \models_\upsilon \phi$ *or* $\mathrm{M_{IS}}, g \models_\upsilon \psi$,
- $\mathrm{M_{IS}}, g \models_\upsilon \langle\langle\Gamma\rangle\rangle X\phi$ *iff for some set* $F_\Gamma$ *of strategies for the agents from* $\Gamma$ *we have that* $\mathrm{M_{IS}}, \lambda_1 \models_\upsilon \phi$ *for each computation* $\lambda \in out(g, F_\Gamma)$,
- $\mathrm{M_{IS}}, g \models_\upsilon \langle\langle\Gamma\rangle\rangle G\phi$ *iff for some set* $F_\Gamma$ *of strategies for the agents from* $\Gamma$ *for each computation* $\lambda \in out(g, F_\Gamma)$ *we have* $\mathrm{M_{IS}}, \lambda_i \models_\upsilon \phi$ *for all* $i \in \mathbb{N}$,
- $\mathrm{M_{IS}}, g \models_\upsilon \langle\langle\Gamma\rangle\rangle \phi U \psi$ *iff there exists a set* $F_\Gamma$ *of strategies for the agents from* $\Gamma$ *such that for each computation* $\lambda \in out(g, F_\Gamma)$ *there exists a position* $i \in \mathbb{N}$ *such that* $\mathrm{M_{IS}}, \lambda_i \models_\upsilon \psi$ *and* $\mathrm{M_{IS}}, \lambda_j \models_\upsilon \phi$ *for each* $0 \leq j < i$,

- $\mathrm{M}_{\mathrm{IS}}, g \models_\upsilon \langle\langle Y \rangle\rangle \xi$ *iff* $\mathrm{M}_{\mathrm{IS}}, g \models_\upsilon \langle\langle \upsilon(Y) \rangle\rangle \xi$, *for* $\xi \in \{X\phi, G\phi, \phi U \psi\}$.

*where* $\phi, \psi$ *are ATLP formulae,* $Y \in \mathrm{GVars}$, *and* $\Gamma \subseteq \mathrm{Agents}$.

We omit the model symbol, writing $g \models_\upsilon \phi$, if the model over which we interpret the formula $\phi$ is clear from the context. If the formula $\phi$ does not contain parameters (i.e., belongs to ATL), then its validity does not depend on a group assignment. In this case we can simply write $g \models \phi$.

## 3 Symbolic Model Checking for ATLP

Throughout this section assume that $\mathrm{IS} = \left\langle (L_i, Act_i, P_i, t_i)_{i \in \mathrm{Agents} \cup \{E\}}, I, \mathcal{L} \right\rangle$ is an interpreted system, $\mathrm{M}_{\mathrm{IS}} = (G, t, \mathcal{L})$ is its model, and GVars is a fixed set of parameters. We aim at characterizing automatically the set of group assignments that make the given ATLP formula true in a given state of the model. Let $\phi \in \mathrm{ATLP}$ and define the *characteristic function* $f_\phi : G \to 2^{\mathrm{GroupVals}}$ such that $g \models_\upsilon \phi$ iff $\upsilon \in f_\phi(g)$. In what follows we present how to compute $f_\phi$ efficiently.

### 3.1 Boolean Connectives and Non-parametric Modalities

We begin with the Boolean connectives, then discuss the non-parametric modalities and finally present our approach for the parametric modalities.

In the examples presented in this section, we consider only formulae containing a single parameter $Y$. We write $\{Y \rightsquigarrow \{A_1, \ldots, A_m\}\}$ to represent the set of assignments $\{\upsilon_1, \ldots, \upsilon_m\}$ s.t. $\upsilon_i(Y) = A_i$, for all $1 \leq i \leq m$.

*Propositional Variables*
If $p \in \mathcal{PV}$ is a propositional variable, then:

$$f_p(g) = \begin{cases} \mathrm{GroupVals} & \text{if } p \in \mathcal{L}(g), \\ \emptyset & \text{otherwise.} \end{cases}$$

In this case $g \models_\upsilon p$ does not depend upon the group assignment $\upsilon$. Thus, $p$ holds for any $\upsilon$ in the marked states and only there.

*Negation*
Let $f : G \to 2^{\mathrm{GroupVals}}$, then $\overline{f} : G \to 2^{\mathrm{GroupVals}}$ is such a function that $\overline{f}(g) = \mathrm{GroupVals} \setminus f(g)$ for each $g \in G$. It is easy to see that $f_{\neg\phi} = \overline{f_\phi}$.

*Example 2 (Negation).* In the model from Example 1 we have:

$$\overline{f_p}(g) = \begin{cases} \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\}\} & \text{if } g = w_0, \\ \emptyset & \text{for } g \in \{w_1, w_2\}. \end{cases}$$

*Disjunction*
Let $\phi, \psi$ be formulae of ATLP. Then, we have $f_{\phi \vee \psi}(g) = f_\phi(g) \cup f_\psi(g)$ for each state $g \in G$. This follows easily from the fact that $g \models_\upsilon \phi \vee \psi$ iff $g \models_\upsilon \phi$ or $g \models_\upsilon \psi$.

*Non-parametric Modalities*
We tackle the following problem: given $f_\phi$ (and $f_\psi$ in the case of *until*) and $\Gamma \subseteq \text{Agents}$, compute $f_{\langle\langle\Gamma\rangle\rangle X\phi}$, $f_{\langle\langle\Gamma\rangle\rangle G\phi}$, and $f_{\langle\langle\Gamma\rangle\rangle \phi U \psi}$.

Before we proceed, we need to introduce some auxiliary notions. Let $g, g' \in G$ and $\Gamma \subseteq \text{Agents}$. The set of such joint actions for the agents in $\Gamma$ that can be extended to some joint action resulting in a transition from $g$ to $g'$ is defined as follows:

$$linkact_\Gamma(g, g') = \{a \in P_\Gamma(g) \mid \exists_{b \in P_{\text{Agents}\setminus\Gamma}(g)}\ g \overset{(a,b)}{\to} g'\}.$$

*Example 3.* Let us consider the interpreted system presented in Figure 1 and consider the state $w_1$. We have $linkact_{\{1\}}(w_1, w_1) = linkact_{\{2\}}(w_1, w_1) = \emptyset$ and $linkact_{\{1\}}(w_1, w_2) = linkact_{\{1\}}(w_1, w_0) = \{a, b\}$ and $linkact_{\{2\}}(w_1, w_2) = \{x\}$ and $linkact_{\{2\}}(w_1, w_0) = \{y\}$.

The next algorithm is the foundation of our approach to group synthesis for ATLP formulae. As we show in Lemma 1, $f_{\langle\langle\Gamma\rangle\rangle X\phi} = \text{Synth}_\text{X}(f_\phi, \Gamma)$.

**Algorithm 1** $\text{Synth}_\text{X}(f, \Gamma)$

**Input:** $f \in \left(2^{\text{GroupVals}}\right)^G$, $\Gamma \subseteq \text{Agents}$
**Output:** $h \in \left(2^{\text{GroupVals}}\right)^G$

1: $negpairs := \emptyset$, $pairs := \emptyset$, $result := \emptyset$
2: **for all** $g \in G$ **do**
3: $\quad negpairs(g) := \bigcup_{g' \in G} linkact_\Gamma(g, g') \times \overline{f}(g')$
4: $\quad pairs(g) := (P_\Gamma(g) \times \text{GroupVals}) \setminus negpairs(g)$
5: $\quad result(g) := \{\upsilon \mid \exists_a (a, \upsilon) \in pairs(g)\}$
6: **end for**
7: **return** *result*

We apply Algorithm 1 to the model in Figure 1.

*Example 4 (Parametric Preimage).* In Example 2 we calculated $\overline{f}_p$ (i.e., $f_{\neg p}$). Let us focus on the case of $\text{Synth}_\text{X}(f_p, \{2\})$ and perform a single run of the loop 2–6 of Algorithm 1 for $g = w_1$. From Example 3 we have:

$$\begin{aligned} negpairs(w_1) &= linkact_{\{2\}}(w_1, w_1) \times \overline{f_p}(w_1) \cup linkact_{\{2\}}(w_1, w_0) \times \overline{f_p}(w_0) \\ &\quad \cup linkact_{\{2\}}(w_1, w_2) \times \overline{f_p}(w_2) \\ &= \emptyset \times \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\} \cup \{y\} \times \{Y \rightsquigarrow \{\{1\}, \{2\}, \{1,2\}\} \\ &\quad \cup \{a, b\} \times \emptyset \\ &= \{y\} \times \text{GroupVals}. \end{aligned}$$

It is easy to verify that $negpairs(w_1)$ consists of all pairs $(\xi, \upsilon) \in P_{\{2\}}(w_1) \times \text{GroupVals}$ such that for some $\eta \in P_{\{1\}}(w_1)$ and some state $w'$ we have $w_1 \overset{(\eta,\xi)}{\to} w'$ and $w' \not\models_\upsilon p$. This means that for each group assignment, if the agent 2 selects

the action $y$ in state $w_1$, then agent 1 can ensure that in the next step $p$ does not hold. With this in mind, notice that (Line 4):

$$\begin{aligned} pairs(w_1) &= (P_{\{2\}}(w_1) \times \text{GroupVals}) \setminus negpairs(w_1) \\ &= \{x, y\} \times \text{GroupVals} \setminus \{y\} \times \text{GroupVals} \\ &= \{x\} \times \text{GroupVals}. \end{aligned}$$

The $pairs(w_1)$ set is therefore the complement of $negpairs(w_1)$, i.e., a set of such pairs $(\xi', \upsilon')$ that if agent 2 selects action $\xi'$ then agent 1 cannot avoid that in the next step $p$ holds. Now it suffices to forfeit the action symbol to obtain (Line 5) that $result(w_1) = \text{GroupVals}$. Indeed, notice that $w_1 \models_\upsilon \langle\langle 2 \rangle\rangle Xp$ for all $\upsilon \in \text{GroupVals}$.

**Lemma 1.** *For all $\phi \in ATLP$, $\Gamma \subseteq \text{Agents}$, $g \in G$, and $\upsilon \in \text{GroupVals}$:*

$$\text{M}_{\text{IS}}, g \models_\upsilon \langle\langle \Gamma \rangle\rangle X\phi \text{ iff } \upsilon \in \text{Synth}_{\text{X}}(f_\phi, \Gamma)(g).$$

*Proof.* Let us fix $g \in G$ and recall that $\overline{f_\phi} = f_{\neg\phi}$. For a given $g' \in G$ we have:

$$linkact_\Gamma(g, g') \times f_{\neg\phi}(g') = \{(a, \upsilon) \mid \exists_{b \in P_{\text{Agents}\setminus\Gamma}(g)} \, (g \xrightarrow{(a,b)} g' \wedge g' \not\models_\upsilon \phi)\}.$$

Combined with the union over all $g' \in G$ we obtain (Line 3):

$$negpairs(g) = \{(a, \upsilon) \mid \exists_{g' \in G} \exists_{b \in P_{\text{Agents}\setminus\Gamma}(g)} \, (g \xrightarrow{(a,b)} g' \wedge g' \not\models_\upsilon \phi)\}.$$

It follows that (Line 4):

$$(P_\Gamma(g) \times \text{GroupVals}) \setminus negpairs(g) = \{(a, \upsilon) \mid \forall_{b \in P_{\text{Agents}\setminus\Gamma}(g)} (g \xrightarrow{(a,b)} g' \Longrightarrow g' \models_\upsilon \phi)\}$$

therefore it suffices (Line 5) to forfeit the explicit action to obtain the correct result. □

Let us move to the case of the *globally* modality. As presented in the following lemma, we have that $f_{\langle\langle\Gamma\rangle\rangle G\phi} = \text{Synth}_{\text{G}}(f_\phi, \Gamma)$.

**Algorithm 2** $\text{Synth}_{\text{G}}(f_\phi, \Gamma)$

**Input:** $f_\phi \in \left(2^{\text{GroupVals}}\right)^G$, $\Gamma \subseteq \text{Agents}$
**Output:** $f_{\langle\langle Y \rangle\rangle G\phi} \in \left(2^{\text{GroupVals}}\right)^G$
1: $f := \emptyset$, $h := f_\phi$
2: **while** $f \neq h$ **do**
3: $\quad f := h$
4: $\quad h := \text{Synth}_{\text{X}}(f, \Gamma) \cap f_\phi$
5: **end while**
6: **return** $h$

**Lemma 2.** *For all $\phi \in ATLP$, $\Gamma \subseteq$ Agents, $g \in G$, and $\upsilon \in$ GroupVals:*

$$\mathrm{M}_{\mathrm{IS}}, g \models_{\upsilon} \langle\langle \Gamma \rangle\rangle G\phi \text{ iff } \upsilon \in \mathrm{Synth}_{\mathrm{G}}(f_{\phi}, \Gamma)(g).$$

*Proof.* (Sketch) Let us fix a state $g \in G$ and for each $i \in \mathbb{N}$ denote by $h_i$ the value of the variable $h$ after the $i$–th run of the loop between lines 2 and 5 (assume $h_0 = f_{\phi}$). Firstly, by straightforward induction, with the assumption on locally defined strategies employed during an inductive step, we prove that:

$$h_i(g) = \{\upsilon \mid \exists_{F_{\Gamma}} \forall_{\lambda \in out(g, F_{\Gamma})} \forall_{0 \leq j \leq i}\ \lambda_j \models_{\upsilon} \phi\}$$

for each $i \geq 0$. Intuitively, $h_i(g)$ consists of such group assignments that $\phi$ can be enforced by $\Gamma$ up to the depth $i$. To conclude, it suffices to notice that for each $g \in G$ the sets $\{h_i(g)\}_{i \in \mathbb{N}}$ form a descending chain, thus the while loop 2 – 5 terminates on the fixpoint $h$, satisfying $h(g) = \bigcap_{i=0}^{\infty} h_i(g)$. □

The next algorithm is also similar to its non-parametric counterpart. As shown in Lemma 3 we have $f_{\langle\langle \Gamma \rangle\rangle \phi U \psi} = \mathrm{Synth}_{\mathrm{U}}(f_{\phi}, f\psi, \Gamma)$.

**Algorithm 3** $\mathrm{Synth}_{\mathrm{U}}\,(f_{\phi}, f_{\psi}, \Gamma)$

**Input:** $f_{\phi} \in \left(2^{\mathrm{GroupVals}}\right)^{G}$, $f_{\psi} \in \left(2^{\mathrm{GroupVals}}\right)^{G}$, $\Gamma \subseteq$ Agents
**Output:** $f_{\langle\langle Y \rangle\rangle \phi U \psi} \in \left(2^{\mathrm{GroupVals}}\right)^{G}$

1: $f := G \times \mathrm{GroupVals}$, $h := f_{\psi}$
2: **while** $f \neq h$ **do**
3: $\quad f := h$
4: $\quad h := f_{\psi} \cup (\mathrm{Synth}_{\mathrm{X}}(f, \Gamma) \cap f_{\phi})$
5: **end while**
6: **return** $h$

**Lemma 3.** *For all $\phi, \psi \in ATLP$, $\Gamma \subseteq$ Agents, $g \in G$ and $\upsilon \in$ GroupVals:*

$$\mathrm{M}_{\mathrm{IS}}, g \models_{\upsilon} \langle\langle \Gamma \rangle\rangle \phi U \psi \text{ iff } \upsilon \in \mathrm{Synth}_{\mathrm{G}}(f_{\phi}, f_{\psi}, \Gamma)(g).$$

*Proof.* (Sketch) Similarly as in the case of Lemma 2 we fix a state $g \in G$ and for each $i \in \mathbb{N}$ denote by $h_i$ the value of the variable $h$ after the $i$–th run of the 2 – 5 loop, assuming $h_0 = f_{\phi}$. By induction we prove that:

$$h_i(g) = \{\upsilon \mid \exists_{F_{\Gamma}} \forall_{\lambda \in out(g, F_{\Gamma})} (\exists_{j \leq i}\ \lambda_j \models_{\upsilon} \psi \wedge \forall_{0 \leq k < j}\ \lambda_k \models_{\upsilon} \phi)\}$$

for all $i \in \mathbb{N}$. Intuitively, this means that $h_i(g)$ consists of all such group assignments $\upsilon$ that $\Gamma$ can enforce that along each outcome, a state in which $\psi$ holds under $\upsilon$ will be present, in not later than the $i$–th position from the beginning, while $\phi$ holds under $\upsilon$ in all earlier positions. Notice that the algorithm stops as for each $g \in G$ the sets $\{h_i(g)\}_{i \in \mathbb{N}}$ form an ascending chain in the finite domain, thus the 2 – 5 loop stops on the fixpoint $h$ such that $h(g) = \bigcup_{i=0}^{\infty} h_i(g)$. □

### 3.2 Parametric Modalities

To compute $f_{\langle\langle Y\rangle\rangle\xi}$, where $\xi \in \{X\phi, G\phi, \phi U\psi\}$ we employ an explicit enumeration over the subsets of Agents.

Let $f : G \to 2^{\text{GroupVals}}$ be a function; we define a *restriction* of $f$ with respect to $Y = \Gamma$ as the function $f_{|Y=\Gamma} : G \to 2^{\text{GroupVals}}$ such that $f_{|Y=\Gamma}(g) = \{\upsilon \in f(g) \mid \upsilon(Y) = \Gamma\}$ for $g \in G$.

**Algorithm 4** pSynth $(f_\phi, f_\psi, Y, \mathit{modality})$

**Input:** $f_\phi \in \left(2^{\text{GroupVals}}\right)^G$, $f_\psi \in \left(2^{\text{GroupVals}}\right)^G$, $Y \in$ GVars, $\mathit{modality} \in \{X, G, U\}$
**Output:** $h \in \left(2^{\text{GroupVals}}\right)^G$

```
h := ∅
for all Γ ⊆ Agents, Γ ≠ ∅ do
    switch (modality)
    case modality = X:
        h := h ∪ Synth_X(f_φ, Γ)_{|Y=Γ}
    case modality = G:
        h := h ∪ Synth_G(f_φ, Γ)_{|Y=Γ}
    case modality = U:
        h := h ∪ Synth_U(f_φ, f_ψ, Γ)_{|Y=Γ}
    end switch
end for
return h
```

From Lemma 1, 2, and 3 it follows that $f_{\langle\langle Y\rangle\rangle X\phi} = \text{pSynth}(f_\phi, *, Y, X)$, $f_{\langle\langle Y\rangle\rangle G\phi} = \text{pSynth}(f_\phi, *, Y, G)$, and $f_{\langle\langle Y\rangle\rangle \phi U\psi} = \text{pSynth}(f_\phi, f_\psi, Y, U)$, i.e., the following lemma holds.

**Lemma 4.** *Let $Y \in$ GVars be a parameter. For all $\phi \in ATLP$, $\Gamma \subseteq$ Agents, $g \in G$, and $\upsilon \in$ GroupVals:*

- $\text{M}_{\text{IS}}, g \models_\upsilon \langle\langle Y\rangle\rangle X\phi$ iff $\upsilon \in \text{pSynth}(f_\phi, *, Y, X)(g)$,
- $\text{M}_{\text{IS}}, g \models_\upsilon \langle\langle Y\rangle\rangle G\phi$ iff $\upsilon \in \text{pSynth}(f_\phi, *, Y, G)(g)$,
- $\text{M}_{\text{IS}}, g \models_\upsilon \langle\langle Y\rangle\rangle \phi U\psi$ iff $\upsilon \in \text{pSynth}(f_\phi, f_\psi, Y, U)(g)$,

*for all $g \in G$ and all group assignments $\upsilon$.*

*Overall Algorithm*
The following algorithm summarises the cases presented earlier, providing the overall algorithm for the group synthesis for ATLP formulae.

**Algorithm 5** $\mathrm{Synth}_{\mathrm{ATLP}}(\phi)$

**Input:** $\phi \in$ ATLP
**Output:** $f_\phi \in \left(2^{\mathrm{GroupVals}}\right)^G$

1: **if** $\phi = \langle\!\langle Y \rangle\!\rangle \mathcal{Z}\psi$ **then**
2: **return** $\mathrm{pSynth}(\mathrm{Synth}_{\mathrm{ATLP}}(\psi), *, Y, \mathcal{Z})$
3: **else if** $\phi = \langle\!\langle Y \rangle\!\rangle \psi U \xi$ **then**
4: **return** $\mathrm{pSynth}(\mathrm{Synth}_{\mathrm{ATLP}}(\psi), \mathrm{Synth}_{\mathrm{ATLP}}(\xi), Y, U)$
5: **else** {propositional and non-parametric modalities omitted for simplicity}
6: **return** $f_\phi$
7: **end if**

The validity of the next theorem follows directly from our previous analysis of propositions, boolean and non-parametric operations, and Lemma 4.

**Theorem 1 (Group Synthesis for ATLP).**
*For all $\phi \in ATLP$, $\Gamma \subseteq$ Agents, $g \in G$, and $\upsilon \in$ GroupVals:*

$$\mathrm{M}_{\mathrm{IS}}, g \models_\upsilon \phi \text{ iff } \upsilon \in \mathrm{Synth}_{\mathrm{ATLP}}(\phi)(g).$$

In Algorithm 5 the procedures $\mathrm{Synth}_{\mathrm{X}}$, $\mathrm{Synth}_{\mathrm{G}}$ and $\mathrm{Synth}_{\mathrm{U}}$ are indirectly called as subroutines by Algorithm 4. As these operate on functions (standard ATL verification algorithms work with sets of states) the application of pSynth to formulae with $k$ nested parametric modalities requires $2^{|\mathrm{Agents}|+k-1}$ calls. The brute-force approach performs in this case $2^{k|\mathrm{Agents}|}$ calls to non-parametric ATL verification procedures. Note that while both the approaches are exponential with respect to the number of calls of fixed-point subroutines, the parametric approach admits exponentially less calls than the brute-force one. The practical benefits of this are very substantial, as demonstrated by the preliminary experimental results presented in the next section.

## 4 Experimental Evaluation

We have implemented the presented theory as an experimental extension to the open-source multi-agent systems model checker MCMAS [13]. A GNU-GPL licenced release of the toolkit is available from [1]. As MCMAS is a symbolic model checker based on BDDs, we encode sets of groups and perform synthesis symbolically.

In the experimental evaluation $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in$ GVars are parameters; we also use the following standard abbreviations:

- $\langle\!\langle \mathcal{X} \rangle\!\rangle F\phi \triangleq \langle\!\langle \mathcal{X} \rangle\!\rangle [true U \phi]$
- $[\![\mathcal{X}]\!] F\phi \triangleq \neg\langle\!\langle \mathcal{X} \rangle\!\rangle G\neg\phi$
- $[\![\mathcal{X}]\!] G\phi \triangleq \neg\langle\!\langle \mathcal{X} \rangle\!\rangle F\neg\phi$

To exemplify, $\langle\!\langle \mathcal{X} \rangle\!\rangle F\phi$ is read "the group $\mathcal{X}$ can enforce a future where $\phi$ holds", and $[\![\mathcal{X}]\!] G\phi$ is read "the group $\mathcal{X}$ cannot avoid $\phi$ holding globally".

To evaluate the efficiency of the proposed solution to the group synthesis problem, we compare the parametric approach to the brute-force one that iteratively checks all possible substitutions for all groups variables. For the brute-force approach, checking a formula with $m$ group variables over a model with $n$ agents requires checking $(2^n - 1)^m$ possible assignments.

The values presented are the average over three runs, with a timeout of one hour per run (set using `` `timeout 1h` ``). The machine employed for these benchmarks was an Intel Core i5 processor 3.20 GHz, with a 4,096 KiB cache running 32-bit Fedora 16, kernel 3.6.11-4. The presented values were obtained using the TSTIME tool; time represents the 'real' CPU time while memory represents the "high-water mark" of allocated memory as reported by the Linux kernel.

We evaluate the proposed technique on two benchmarks. The first (Section 4.1) is an industrial benchmark, which we use to demonstrate a real-world applicability. The second (Section 4.2) is an academic benchmark, but is scalable to allow us to effectively benchmark the parametric approach over various sized models and show the possible gains.

### 4.1 IEEE Token Ring Network with Faults

We first compare the parametric and brute-force approaches using the industry standard IEEE token ring bus network. In the comparison that follows, we automated the injection of faults into the model, following the approach of [7]. We briefly summarise the scenario below; for a complete description we refer the reader to [7].

The IEEE token ring protocol connects $k$ nodes in a ring topology; data moves among nodes on the network in a clockwise fashion. Access is granted to nodes on the network in the form of a token; this is passed from node to node. Tokens are issued onto the network from an "active monitor". To detect faults, tokens contain a "time to live" field, initialised to the maximum time that a token would take to circulate the whole network and counting down to zero. Should a token fail to circulate back to the active monitor within the given time-frame, it is deduced that a fault has occurred on the network.

We consider instantiations of the network where the first node wishes to transmit a data token to the final node. Consequently, data needs to pass through every single intermediate node on the system.

Using a modified version of the fault injector from [7], we inserted non-deterministic "state replace" faults into each node, which causes the node to switch from a *send* state to a *disconnected* state. Each fault inserted into the system has a corresponding "fault injector" agent that can trigger the fault.

For a token ring network with $k$ nodes, we synthesise the groups for the following specifications:

- $\phi_{\text{TR1}} = \langle\!\langle \mathcal{X} \rangle\!\rangle F \left( \bigwedge_{i \in 1 \ldots k} \mathit{disconnected}_i \right)$
- $\phi_{\text{TR2}} = \langle\!\langle \mathcal{X} \rangle\!\rangle \left[ \left( [\![ \mathcal{Y} ]\!] F \left( \bigwedge_{i \in 1 \ldots k} \mathit{disconnected}_i \right) \right) U \left( \bigwedge_{i \in 1 \ldots k} \mathit{disconnected}_i \right) \right]$
- $\phi_{\text{TR3}} = \langle\!\langle \mathcal{X} \rangle\!\rangle F \left( [\![ \mathcal{Y} ]\!] G \left( \langle\!\langle \mathcal{Z} \rangle\!\rangle X \left( \bigwedge_{i \in 1 \ldots k} \mathit{disconnected}_i \right) \right) \right)$

where the proposition $disconnected_i$ holds in a state if the $i$-th node is in a disconnected state.

Under a ground interpretation (i.e., reading the parametric modalities as their non-parametric equivalents), the formulae are read as follows: (i) $\phi_{\text{TR1}}$ states "the group $\mathcal{X}$ can enforce a future state where all the nodes are disconnected"; (ii) $\phi_{\text{TR2}}$ expresses "the group $\mathcal{X}$ can enforce a future state where all the nodes are disconnected and, until that point, the group $\mathcal{Y}$ cannot avoid that eventuality"; and (iii) $\phi_{\text{TR3}}$ denotes "the group $\mathcal{X}$ can enforce a future where the group $\mathcal{Y}$ cannot avoid globally that the group $\mathcal{Z}$ can enforce that all the nodes are disconnected at the next state".

Table 1: Comparison for the Token Ring Network

| Model | | Formula | Group Valuations | | Time (s) | | Memory (KiB) | |
|---|---|---|---|---|---|---|---|---|
| Nodes | States | | Possible | # SAT | PARAMETRIC | BRUTE-FORCE | PARAMETRIC | BRUTE-FORCE |
| 2 | 9,260 | $\varphi_{\text{TR1}}$ | 15 | 4 | 3.818 | 3.728 | 24,072 | 24,464 |
| | | $\varphi_{\text{TR2}}$ | 225 | 12 | 6.369 | 49.635 | 24,096 | 24,856 |
| | | $\varphi_{\text{TR3}}$ | 3,375 | 900 | 8.836 | 425.677 | 24,172 | 24,132 |
| 3 | 260,797 | $\varphi_{\text{TR1}}$ | 63 | 8 | 1573.030 | 1597.111 | 65,452 | 65,092 |
| | | $\varphi_{\text{TR2}}$ | 3,969 | – | – | – | – | – |
| | | $\varphi_{\text{TR3}}$ | 250,047 | – | – | – | – | – |

*Discussion of Results.* Table 1 shows the comparison between the parametric and brute-force approach. These results demonstrate the speed benefits of using parametric verification.

Unsurprisingly, for $\varphi_{\text{TR1}}$ the smallest synthesised substitution for $\mathcal{X}$ is the group consisting of all of the fault injectors, as only in collaboration can the fault injectors cause every node on the ring to become disconnected. Similarly, for $\varphi_{\text{TR2}}$ we obtain the fault injectors for $\mathcal{X}$ and all the nodes for $\mathcal{Y}$, as they are powerless to stop themselves becoming disconnected once the fault injectors form a coalition.

For a model with 3 nodes (and, therefore, 3 fault injectors) and a formula with more than one parameter, neither tool was able to synthesise a result within the one hour timeout.

### 4.2 Generic Pipeline

We now consider a scalable, multi-agent systems version of the generic pipeline paradigm [16]. The model contains a producer, a consumer, and a chain of $k$ intermediate processing nodes.

The producer can be in one of two states: idle and ready-to-feed. When it is in the ready-to-feed state, it can pass a product to the first intermediate node in the chain. The producer stays in the ready-to-feed state for up to two transitions,

unless the first node accepts the product and so the producer changes to idle. The producer can idle at most two turns; then it returns to being ready-to-feed.

Unlike the producer, each intermediate node can alternate between the ready-to-eat and ready-to-feed states. There is no time limit for a node to stay ready-to-eat, but it has to accept a product offered by its predecessor. Upon accepting, the node changes to ready-to-feed and stays there for two turns, unless its successor accepts the passed product.

The consumer constitutes the end of the pipeline; it can stay in any of its states, idle or ready-to-eat, for at most two transitions. When in ready-to-eat, the consumer is ready to accept a product from the last node of the chain.

For a pipeline with $k$ intermediate nodes, we synthesise groups for the following properties:

- $\phi_{\text{GP1}} = \langle\!\langle \mathcal{X} \rangle\!\rangle F \left( idle_{consumer} \wedge idle_{producer} \right)$
- $\phi_{\text{GP2}} = \langle\!\langle \mathcal{X} \rangle\!\rangle [(\llbracket \mathcal{Y} \rrbracket F \left( rtf_1 \wedge rtf_k \right)) \, U \, (rtf_1 \wedge rtf_k)]$
- $\phi_{\text{GP3}} = \langle\!\langle \mathcal{X} \rangle\!\rangle F \left( \llbracket \mathcal{Y} \rrbracket G \left( \langle\!\langle \mathcal{Z} \rangle\!\rangle X \left( rtf_{producer} \wedge rte_1 \right) \vee \left( rtf_k \wedge rte_{consumer} \right) \right) \right)$

where the propositions $idle_i$ holds in a global state if the $i$-th node (similarly *producer* for the producer and *consumer* for the consumer) is idle; $rtf_i$ holds in a global state if the $i$-th node is ready-to-feed; and $rte_i$ holds in a global state if the $i$-th node is ready-to-eat. The properties have similar interpretations to those in the token ring network of Section 4.1.

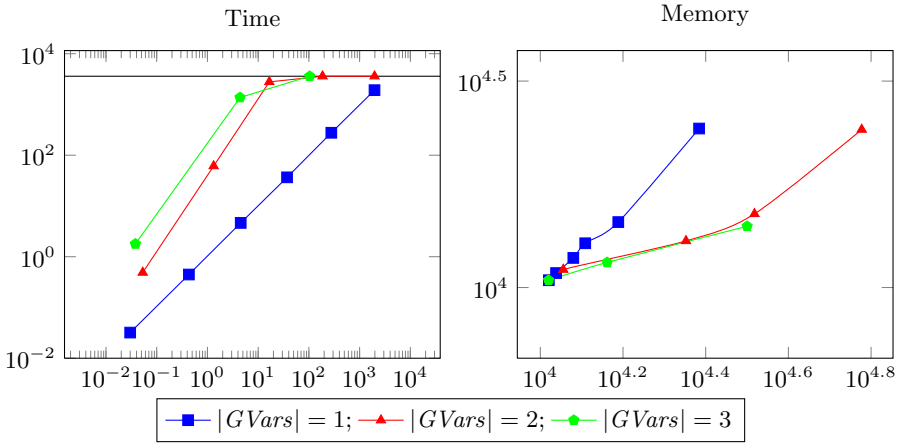


Figure 2: Comparison between parametric and brute-force (x-axis parametric, y-axis brute-force).

*Discussion of Results.* The results of parametric synthesis for various sizes of pipeline is shown in Figure 2.

The time graph shows a comparison between parametric and brute-force; the horizontal solid black line represents the timeout period of 3,600 seconds. This graph clearly shows that as we increase the number of parameters in the formula, the approach presented out-performs the brute-force approach.

The memory graph demonstrates the speed-up is not without cost: the parametric approach requires a higher memory overhead to store the results of the intermediate checks. This is to be expected.

## 5 Conclusions and Future Work

There has been considerable interest recently in verifying agent-based specifications by means of model checking. ATL is a key formalism to represent strategic abilities of agents in a system; several approaches have been put forward to verify ATL specifications [2,4,12]. In this paper we introduced a parametric variant to ATL and introduced the problem of synthesising the groups of agents satisfying a parametric specification.

We put forward parameter synthesis algorithms to synthesise the groups and proved their correctness. We showed they are amenable to be implemented in combinations with BDDs and introduced a model checker that implements the procedures. The experimental results obtained show that the technique can, in general, speed up considerably the group synthesis problem, compared to checking all possible groups for the specification in question. As expected the experiments showed no improvement on the memory footprint of the procedure against the brute-force approach.

While we already demonstrated the technique on an industrial use case from the networking literature in the future we plan to use the technique in the context of strategically controlling the actions of robots in a team to achieve a given set of objectives.

## References


1. MCMAS-ATLP. `http://vas.doc.ic.ac.uk/software/tools/` (2013)
2. Alur, R., de Alfaro, L., Grosu, R., Henzinger, T.A., Kang, M., Kirsch, C.M., Majumdar, R., Mang, F.Y.C., Wang, B.Y.: JMOCHA: A Model Checking Tool that Exploits Design Structure. In: Proc. of ICSE'01. pp. 835–836 (2001)
3. Alur, R., Etessami, K., Torre, S.L., Peled, D.: Parametric Temporal Logic for "Model Measuring". ACM Trans. Comput. Log. 2(3), 388–407 (2001)
4. Alur, R., Henzinger, T., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM 49(5), 672–713 (2002)
5. Brihaye, T., Markey, N., Ghannem, M., Rieg, L.: Good Friends are Hard to Find! In: Proc. of TIME'08. pp. 32–40 (2008)
6. Clancey, W.J., Sierhuis, M., Seah, C., Buckley, C., Reynolds, F., Hall, T., Scott, M.: Multi-Agent Simulation to Implementation: A Practical Engineering Methodology for Designing Space Flight Operations. In: Proc. of ESAW'07. pp. 108–123 (2007)
7. Ezekiel, J., Lomuscio, A.: A Methodology for Automatic Diagnosability Analysis. In: Proc. of ICFEM'10. pp. 549–564 (2010)

8. Gascueña, J.M., Fernández-Caballero, A.: Review: on the Use of Agent Technology in Intelligent, Multisensory and Distributed Surveillance. Knowl. Eng. Rev. 26(2), 191–208 (2011)
9. Himoff, J., Skobelev, P., Wooldridge, M.: MAGENTA Technology: Multi-Agent Systems for Industrial Logistics. In: Proc. of AAMAS'05. pp. 60–66 (2005)
10. Jacobi, S., Madrigal-Mora, C., León-Soto, E., Fischer, K.: AgentSteel: An Agent-based Online System for the Planning and Observation of Steel Sroduction. In: Proc. of AAMAS'05. pp. 114–119 (2005)
11. Jones, A.V., Knapik, M., Lomuscio, A., Penczek, W.: Group Synthesis for Parametric Temporal-Epistemic Logic. In: Proc. of AAMAS'12. pp. 1107–1114 (2012)
12. Kacprzak, M., Penczek, W.: Fully Symbolic Unbounded Model Checking for Alternating-time Temporal Logic. AAMAS 11(1), 69–89 (2005)
13. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In: Proc. of CAV'09. pp. 682–688 (2009)
14. Lomuscio, A., Raimondi, F.: Model Checking Knowledge, Strategies, and Games in Multi-Agent Systems. In: Proc. of AAMAS'06. pp. 161–168 (2006)
15. Monica, D.D., Napoli, M., Parente, M.: On a Logic for Coalitional Games with Priced-Resource Agents. Electron. Notes Theor. Comput. Sci. 278, 215–228 (2011)
16. Peled, D.: All From One, One For All: On Model Checking Using Representatives. In: Proc. of CAV'93. pp. 409–423 (1993)
17. Pěchouček, M., Mařík, V.: Industrial Deployment of Multi-Agent Technologies: Review and Selected Case Studies. AAMAS 17(3), 397–431 (2008)
18. Pěchouček, M., Thompson, S.G., Voos, H.: Defense Industry Applications of Autonomous Agents and Multi-Agent Systems. Birkhauser Basel (2008)
19. Rouff, C.: Autonomous and Autonomic Systems: With Applications to NASA Intelligent Spacecraft Operations and Exploration Systems. Springer-Verlag (2007)
20. Tumer, K., Agogino, A.: Distributed Agent-based Air Traffic Flow Management. In: Proc. of AAMAS'07. pp. 342–349 (2007)