Symbolic Synthesis of Fault-Tolerance Ratios in Parameterised Multi-Agent Systems

Panagiotis Kouvaros[1], **Alessio Lomuscio**[2] and **Edoardo Pirovano**[2]
[1] University of Cyprus, Department of Computer Science
[2] Imperial College London, Department of Computing
panagiotis.kouvaros@gmail.com, a.lomuscio@ic.ac.uk, e.pirovano17@ic.ac.uk

Abstract

We study the problem of determining the robustness of a multi-agent system of unbounded size against specifications expressed in a temporal-epistemic logic. We introduce a procedure to synthesise automatically the maximal ratio of faulty agents that may be present at runtime for a specification to be satisfied in a multi-agent system. We show the procedure to be sound and amenable to symbolic implementation. We present an implementation and report the experimental results obtained on a number of protocols from swarm robotics.

1 Introduction

A key criterion when assessing protocols in multi-agent systems (MAS) is establishing their correctness against a set of specifications. While important, this is often not sufficient for deployment in several scenarios of practical significance. Even if a protocol can be shown to be demonstrably correct, the designer has no information on the impact of possible faults and malfunctions at runtime. This is of importance in large MAS where the individual risk of failure is high, as is the case in robotic swarms [Winfield *et al.*, 2005].

It is therefore of interest not just to establish correctness, but the degree of fault-tolerance of a protocol with respect to possible faults in the system. In the case of MAS composed of a fixed number of agents known at design time, techniques inspired by safety-analysis [Bozzano and Villafiorita, 2007], allow the engineer to study the consequences of a variety of faults against specifications of interest [Ezekiel and Lomuscio, 2009; 2017]. For example in [Ezekiel *et al.*, 2011] the resilience of an autonomous submarine AUTOSUB6000 against possible faults was assessed in this manner.

Often, however, the number of agents constituting a MAS is not known in advance, or indeed changes at runtime. This is the case in several areas of artificial intelligence including robotic swarms for monitoring or repair, auctions, etc. In these cases, it is useful to determine the *maximal ratio* of faulty agents (as a proportion of the whole population) that the protocol can tolerate against some specification. For example, in a flocking protocol we may be interested in the percentage of drones that need to be faulty for the rest of the swarm to lose its ability to fly in formation [Kouvaros and Lomuscio, 2017]. Determining this ratio can have significant consequences on engineering aspects of the system.

Under general conditions, determining this ratio is undecidable [Apt and Kozen, 1986]. In [Kouvaros and Lomuscio, 2017], a method for checking a class of systems of interest against a given ratio of faults is put forward. This induces a procedure for calculating an approximation of the ratio. However, this calculation rests on model checking systems of varying sizes and is therefore computationally expensive and often entirely impractical. In this paper we develop a novel technique to derive the fault tolerance ratio symbolically. The ratio, as we explain, is incorporated into a model that encodes various combinations of faulty and correct agents operating at runtime. This symbolic calculation, which amounts to solving a version of the parameterised model checking (PMC) problem [Kouvaros and Lomuscio, 2016] in combination with a generalised model checking algorithm, enables us to synthesise the ratio as a one-shot procedure.

Related work. The PMC problem has been extensively studied [Bloem *et al.*, 2015]. More recently, it has also been used to study fault tolerance in distributed algorithms [Aminof *et al.*, 2018; John *et al.*, 2013; Zhang *et al.*, 2009]. However, these approaches either are not amenable to analysing MAS (due to, for instance, a lack of epistemic specifications) or do not tackle the issue we tackle here of finding the ratio of faulty to non-faulty components.

The rest of the paper is organised as follows. In Section 2 we fix the notation and introduce the parameterised semantics upon which the framework is developed. In Section 3 we present a generalised model checking procedure which can be used to synthesise the ratio of faulty agents in a MAS of a fixed number of agents. We build on this in Section 4 where, by combining this with results in parameterised model checking, we derive a provably sound procedure for MAS of arbitrary size. We present an implementation of the technique in Section 5 and report on its use on transport and aggregation protocols for swarm systems. We conclude in Section 6.

2 Background

Models. Parameterised interleaved interpreted systems (PIISs) give a model to reason about MAS composed of an unbounded number of agents by extending interleaved interpreted systems (IIS) [Lomuscio *et al.*, 2010]. The presentation here follows that in [Kouvaros and Lomuscio, 2017]. A PIIS is made up of an *agent template*, which describes the behaviour of individual agents and an *environment* which captures the rest of the state of the system. The framework can accommodate a finite number of different agent templates, but for simplicity we present the definitions with just one.

The agent template $T = \langle L, \iota, Act, P, t\rangle$ defines a finite, non-empty set of local agent states L together with a distinguished initial state $\iota \in L$. The non-empty set $Act = A \cup AE \cup GS$ gives the actions that can be performed by the agents, which are either *asynchronous actions*, *agent-environment actions* or *global-synchronous actions*. Each type of action gives a different communication pattern between the agents, as outlined in Definition 2. The actions are performed in compliance with a protocol $P : L \to \mathcal{P}(Act)$ that defines which actions are enabled in each state. The agent's transition function $t : L \times Act \to L$ describes the evolution of the agent's state: given its local state and the action performed, it returns its new local state.

Similarly, the environment $e = \langle L_e, \iota_e, Act_e, P_e, t_e\rangle$ defines a finite, non-empty set of local states L_e, a distinguished initial state $\iota_e \in L_e$, a non-empty set of actions $Act_e = A_e \cup AE \cup GS$, a protocol $P_e : L_e \to \mathcal{P}(Act_e)$, and a transition function $t_e : L_e \times Act_e \to L_e$.

Definition 1 (PIIS)**.** *A parameterised interleaved interpreted system is a tuple $\mathcal{S} = \langle T, e, \mathcal{V}\rangle$, where $\mathcal{V} : L \to 2^{AP}$ is a labelling function on the agent template's states for a set AP of atomic propositions.*

Each PIIS describes an unbounded collection of concrete systems obtained by choosing different numbers of agents in the system. Given a PIIS $\mathcal{S}$ and $n \in \mathbb{Z}^+$ the IIS $\mathcal{S}(n)$ of n agents is the result of the composition of n copies of T with the environment. We denote the set of concrete agents instantiated from T by $\mathcal{A} = \{1, \ldots, n\}$. A *global state* $g = \langle l_1, \ldots, l_n, l_e\rangle$ is a tuple of local states for all the agents and the environment in $\mathcal{S}(n)$; it describes the system at a particular instant of time. For a global state g we write $g.i$ to denote the local state of agent i in g and $g.e$ to denote the state of the environment in g. The system's global states evolve over time in compliance with the global transition relation.

Definition 2 (Global transition relation)**.** *The global transition relation $R \subseteq G \times Act \cup Act_e \times G$ on a set G of global states is defined as $(g, a, g') \in R$ iff one of the following holds:*

- (Asynchronous). *(i) $a \in A \cup A_e$; (ii) there is $i \in \mathcal{A} \cup \{e\}$ s.t. $a \in P(g.i)$ and $t(g.i, a) = g'.i$; (iii) for all $j \neq i$, $g.j = g'.j$.*
- (Agent-environment). *(i) $a \in AE$; (ii) there is $i \in \mathcal{A}$ s.t. $a \in P(g.i)$ and $t(g.i, a) = g'.i$; (iii) $a \in P_e(g.e)$ and $t_e(g.e, a) = g'.e$; (iv) for all $j \neq i$, $j \neq e$, $g.j = g'.j$.*
- (Global-synchronous). *(i) $a \in GS$; (ii) for all $i \in \mathcal{A}$, $a \in P(g.i)$ and $t(g.i, a) = g'.i$; (iii) $a \in P_e(g.e)$ and $t_e(g.e, a) = g'.e$.*

We sometimes write $g \to g'$ for $\exists a\colon (g, a, g') \in R$.

Above R defines precisely one action to be performed at each time step. If this is an asynchronous action, then exactly one agent participates in the global transition; if it is an agent-environment action, then exactly one agent and the environment participate in the global transition; if it is a global-synchronous action, then all the agents and the environment participate in the global transition.

Having defined the global transition relation, we can proceed to give the concrete semantics describing the behaviour of a system $\mathcal{S}(n)$ composed of n agents and an environment.

Definition 3 (Concrete semantics)**.** *Given a PIIS $\mathcal{S}$ and $n \in \mathbb{Z}^+$, the IIS $\mathcal{S}(n)$ is a tuple $\mathcal{S}(n) = \langle G, g_0, R, V\rangle$, where $G \subseteq L^n \times L_e$ is the set of reachable global states via R from the initial global state $g_0 = \langle \iota, \ldots, \iota, \iota_e\rangle$, and $V : G \to 2^{AP \times \mathcal{A}}$ is the labelling function on the global states defined by $(p, i) \in V(g)$ iff $p \in \mathcal{V}(g.i)$, for each $p \in AP$, $i \in \mathcal{A}$, where AP is a finite set of atomic propositions.*

Notice that for each atomic proposition p we create a copy for each of the n agents and label a global state with (p, i) if the agent i is at a local state labelled with p by the template labelling function.

A *path* π is an infinite sequence $\pi = g^0 a^0 g^1 a^1 g^2 \ldots$ such that $(g^i, a^i, g^{i+1}) \in R$ for every $i \geq 0$. We write $\pi(i)$ for the i-th state in π. The set of all paths originating from a state g is denoted by $\Pi(g)$.

Specifications. We now define the logic ICTLK$\backslash X$ which extends the temporal-epistemic logic CTLK without the next time operator by introducing indexed atomic propositions and indexed epistemic modalities that are quantified over the concrete agents [Kouvaros and Lomuscio, 2016]. Given a set IND of indices, a set AP of atomic propositions, ICTLK$\backslash X$ formulae are defined by the following BNF grammar:

$$\begin{aligned}\phi ::= &\ \text{true} \mid (p, v) \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid E(\phi U \phi)\\ &\mid EG\phi \mid \overline{K}_i\phi \mid \forall v\colon \phi\end{aligned}$$

where $p \in AP$ and $v \in IND$. The epistemic modality $\overline{K}_i\phi$ is read as "agent i regards ϕ as being epistemically possible". The temporal modality $E(\phi_1 U \phi_2)$ denotes "for some path, at some point ϕ_2 holds and before then ϕ_1 is true along the path"; and $EG(\phi)$ denotes "for some path, ϕ holds at every state". We also use standard CTL abbreviations such as $F\phi \equiv \text{true}\ U\phi$. The universal quantifier allows us to express properties irrespective of the number of agents in the system. An ICTLK$\backslash X$ formula is said to be a *sentence* if every variable appearing the formula is in the scope of a universal quantifier. Hereafter we consider only sentences.

Definition 4 (Satisfaction of ICTLK$\backslash X$)**.** *The satisfaction relation $\models$ for an IIS $\mathcal{S}(n)$, and an ICTLK$\backslash X$ sentence ϕ is inductively defined as follows (the clauses for the atomic propositions and the Boolean operators are omitted):*

$(\mathcal{S}(n), g) \models E(\phi_1 U \phi_2)$	*iff*	*for some $\pi \in \Pi(g)$, for some $i \geq 0$ $(\mathcal{S}(n), \pi(i)) \models \phi_2$ and for all $0 \leq j < i$, $(\mathcal{S}(n), \pi(j)) \models \phi_1$;*
$(\mathcal{S}(n), g) \models EG\phi$	*iff*	*for some $\pi \in \Pi(g)$, for all $i \geq 0$, $(\mathcal{S}(n), \pi(i)) \models \phi$;*
$(\mathcal{S}(n), g) \models \overline{K}_i\phi$	*iff*	*for some $g' \in G$ with $g.i = g'.i$ it is the case that $(\mathcal{S}(n), g') \models \phi$;*
$(\mathcal{S}(n), g) \models \forall v\colon \phi$	*iff*	*$(\mathcal{S}(n), g) \models \phi[v \mapsto ag]$ for all $ag \in \mathcal{A}$.*

Notice that, since all agents are behaviourally identical, the evaluation of a universally quantified formula is equivalent to the evaluation of only one of its ground instantiations, where the variables in an instantiation are mapped to different agents [Kouvaros and Lomuscio, 2016]. Hereafter, we will only consider such ground instantiations.

Two fragments of this logic are important for our purposes: the existential fragment ECTLK$\backslash X$ is obtained by restricting negation to only being applied to propositions, i.e. replacing $\neg\phi$ by $\neg(p, v)$ in the syntax. The dual universal fragment ACTLK$\backslash X$ is the negation of this fragment, i.e. the language $\{\neg\phi \mid \phi \in \text{ECTLK} \setminus X\}$.

Our specifications will be expressed in ACTLK$\backslash X$. A formula ϕ is said to be true in $\mathcal{S}(n)$, denoted $\mathcal{S}(n) \models \phi$, if $(\mathcal{S}(n), g_0) \models \phi$. We will use a definition of cutoffs similar to that given in [Kouvaros and Lomuscio, 2013b].

Definition 5 (Cutoff)**.** *An integer $c \in \mathbb{Z}^+$ is said to be a cutoff for a PIIS $\mathcal{S}$ if for all ACTLK$\backslash X$ formulas ϕ we have that $\mathcal{S}(c) \models \phi$ implies $\mathcal{S}(n) \models \phi$ for all $n \geq c$.*

Faults. We will assume that a *faulty* PIIS $\mathcal{S}^f = \langle (T, T^f), e^f, V^f \rangle$ can be constructed from a given PIIS $\mathcal{S} = \langle T, e, V \rangle$ according to the faults we wish to consider. The full details of the faults that can be considered and how the corresponding faulty PIIS is constructed are omitted for space and can be found in [Kouvaros and Lomuscio, 2017]. The faulty PIIS $\mathcal{S}^f$ is constructed from the non-faulty PIIS $\mathcal{S}$ by adding a second agent template T^f that may, in addition to performing all the behaviours T performs, also perform certain faulty behaviours. The environment is modified suitably for synchronisation purposes and the valuation function is extended to label states with a new atomic proposition *faulty* that holds in agent i's local state precisely if agent i has exhibited faulty behaviour at some point in the past. We denote by $\mathcal{S}^f((n_n, n_f))$ the concrete system with n_n non-faulty agents and n_f faulty ones instantiated from T and T^f respectively. Also note that formulae are evaluated on a faulty PIIS by quantifying only over non-faulty agents.

3 Ratio Synthesis

As a stepping stone to parameterised fault tolerance, we now define the notion of fault tolerance for concrete MAS of a fixed size.

Definition 6 (Fault Tolerance)**.** *Given a faulty PIIS $\mathcal{S}^f$, a ratio $\lambda \in [0, 1]$, an integer $n \in \mathbb{Z}^+$ and an ACTLK$\backslash X$ formula ϕ, we say that size n instances of $\mathcal{S}^f$ are λ-tolerant with respect to ϕ if it is the case that:*

$$\mathcal{S}^f((n_n, n_f)) \models \phi \text{ for every } n_n, n_f \in \mathbb{Z}$$

$$\text{such that } n_n + n_f = n \text{ and } n_f/n \leq \lambda$$

If this is the case we write $\mathcal{S}^f \models_n^\lambda \phi$.

Intuitively, the above definition says that if a proportion up to λ of the n agents exhibit faulty behaviour then the specification ϕ is satisfied. In what follows we are concerned with finding how tolerant a system is to faults with respect to a specification. We formalise this decision problem as follows.

Algorithm 1 λ-synthesis Decision Procedure

```
1: procedure $\lambda$-SYNTHESIS( $\mathcal{S}^f, \phi, n$ )
2:     $\mathbb{X} \leftarrow$ Lambda($\mathcal{S}^f((0, n)), \neg\phi$);
3:     if $\exists\rho : (g_0, \rho) \in \mathbb{X}$ then
4:         return $\min\{\rho \in [0, 1] \mid (g_0, \rho) \in \mathbb{X}\}$;
5:     else
6:         return total;
7:     end if
8: end procedure
```

Definition 7 (λ-synthesis)**.** *Given a faulty PIIS $\mathcal{S}^f$, an integer $n \in \mathbb{Z}^+$ and an ACTLK$\backslash X$ formula ϕ, find $\lambda \in [0, 1]$ such that $\mathcal{S}^f \not\models_n^\lambda \phi$ and $\mathcal{S}^f \models_n^{\lambda'} \phi$ whenever $\lambda' < \lambda$, or return* total *if no such λ exists (i.e. the specification is always satisfied).*

Our approach is loosely inspired by the procedures for parameter synthesis used in biological modelling [Beneš *et al.*, 2016; Barnat *et al.*, 2012] but with important adaptations for our setting of fault tolerance in multi-agents systems. Our procedure, shown in Algorithm 1, passes the system of n faulty agents and the *negation* of our specification to a sub-procedure `Lambda`, shown in Algorithm 2.

This sub-procedure works similarly to the labelling algorithm for model checking CTL [Clarke *et al.*, 1999]. The algorithm recursively labels the states of a system with the subformulas of the specification under question that they satisfy. Differently from the CTL labelling algorithm, `Lambda` labels states with both said subformulas and the *minimum* ratio of faulty agents to total agents that is sufficient for the specification to be satisfied at the state. Thus, `Lambda` operates on sets of pairs of states and ratios instead of sets of states. Note that since `Lambda` takes as input the negation of an ACTLK$\backslash X$ formula, the procedure is defined for the existential fragment ECTLK$\backslash X$ of CTLK$\backslash X$.

We now give some preliminary definitions that we will use in our procedure. For a global state g we use $\lambda(g)$ to denote the ratio of agents that have exhibited a fault to total agents, i.e.

$$\lambda(g) \triangleq |\{i \in \mathcal{A} \mid (\text{faulty}, i) \in V(g)\}|/n$$

Given a set of global states X we use $ratio(X)$ to denote the minimum ratio of faulty agents in a state in X. Formally:

$$ratio(X) \triangleq \min\{\lambda(g) \mid g \in X\}$$

For a set $\mathbb{X} \in G \times [0, 1]$ we sometimes simply write X for the projection of $\mathbb{X}$ on the states. Formally:

$$X \triangleq \{g \in G \mid \exists\rho \in [0, 1]\colon (g, \rho) \in \mathbb{X}\}$$

Given $\mathbb{X}$ let $pre_\exists(\mathbb{X})$ denote the pre-image of $\mathbb{X}$ defined as:

$$pre_\exists(\mathbb{X}) \triangleq \{(g, \rho) \mid g \in pre_\exists(X) \wedge \rho = ratio(next(g) \cap X)\}.$$

where $next(g) \triangleq \{g' \mid g \rightarrow g'\}$ expresses the set of successor states of g and $pre_\exists(X) \triangleq \{g \mid \exists g' \colon g \rightarrow g' \wedge g' \in X\}$ denotes the preimage of a set of states X.

Algorithm 2 Labelling Procedure

```
1: procedure LAMBDA( $S^f((0,n)), \phi$ )
2:     case $\phi$
3:         $(p, ag)$ : return $\{(g, \lambda(g)) \mid (p, ag) \in V(g)\}$.
4:         $\neg(p, ag)$ : return $\{(g, \lambda(g)) \mid (p, ag) \notin V(g)\}$.
5:         $\phi_1 \wedge \phi_2$ : return $\mathbb{X}$ where $(g, \rho) \in \mathbb{X}$ iff there are $\rho'$ and $\rho''$ such that $(g, \rho') \in$ Lambda$(S^f((0,n)), \phi_1)$, $(g, \rho'') \in$ Lambda$(S^f((0,n)), \phi_2)$ and $\rho = \max(\rho', \rho'')$.
6:         $\phi_1 \vee \phi_2$ : return $\mathbb{X}$ where $(g, \rho) \in \mathbb{X}$ iff $(g, \rho') \in$ Lambda$(S^f((0,n)), \phi_1)$ or $(g, \rho'') \in$ Lambda$(S^f((0,n)), \phi_2)$, for some $\rho', \rho''$ with $\rho = \min(\rho', \rho'')$.
7:         $\overline{K}_i\psi$ : return Lambda$_{\overline{K}}(S^f((0,n)), i, \psi)$.
8:         $E(\phi_1 U \phi_2)$ :
9:             return Lambda$_{EU}(S^f((0,n)), \phi_1, \phi_2)$.
10:        $EG\psi$ : return Lambda$_{EG}(S^f((0,n)), \psi)$.
11:    end case
12: end procedure
13: procedure LAMBDA$_{\overline{K}}$( $S^f((0,n)), i, \phi$ )
14:    $\mathbb{X} \leftarrow$ Lambda$(S^f((0,n)), \phi)$;
15:    return $\{(g, \rho) \in G \times [0,1] \mid \exists g' \in X : g.i = g'.i \wedge \rho = ratio(\{g' \in X \mid g.i = g'.i\})\}$.
16: end procedure
17: procedure LAMBDA$_{EU}$( $S^f((0,n)), \phi_1, \phi_2$ )
18:    $\mathbb{X} \leftarrow$ Lambda$(S^f((0,n)), \phi_1)$;
19:    $\mathbb{Y} \leftarrow G \times \{0\}$;
20:    $\mathbb{Z} \leftarrow$ Lambda$(S^f((0,n)), \phi_2)$;
21:    while $\mathbb{Y} \neq \mathbb{Z}$ do
22:        $\mathbb{Y} \leftarrow \mathbb{Z}$;
23:        $\mathbb{Z} \leftarrow \mathbb{Z} \cup \{(g, \rho) \in G \times [0,1] \mid \exists \rho', \rho'' \in [0,1] : (g, \rho') \in pre_\exists(\mathbb{Z}) \wedge (g, \rho'') \in \mathbb{X} \wedge \rho = \max(\rho', \rho'')\}$.
24:    end while
25:    return $\mathbb{Z}$;
26: end procedure
27: procedure LAMBDA$_{EG}$( $S^f((0,n)), \phi$ )
28:    $\mathbb{X} \leftarrow G \times \{0\}$;
29:    $\mathbb{Y} \leftarrow$ Lambda$(S^f((0,n)), \phi)$;
30:    while $\mathbb{X} \neq \mathbb{Y}$ do
31:        $\mathbb{X} \leftarrow \mathbb{Y}$;
32:        $\mathbb{Y} \leftarrow \{(g, \rho) \in Y \times [0,1] \mid \exists \rho', \rho'' \in [0,1] : (g, \rho') \in pre_\exists(\mathbb{Y}) \wedge (g, \rho'') \in \mathbb{Y} \wedge \rho = \max(\rho', \rho'')\}$.
33:    end while
34:    return $\mathbb{Y}$;
35: end procedure
```

We now describe the Lambda procedure. The propositional cases are trivial: we can simply label each state which satisfies or does not satisfy the atomic proposition with the ratio of faulty agents at that state. For the conjunction case, notice we need both ϕ_1 and ϕ_2 to hold so the minimum necessary ratio of faults is the largest of the ratios for each of these two to hold. The disjunction case is similar.

For the epistemic modality case, notice we need at least one epistemic successor to satisfy ϕ so we take the minimum ratio for this to be the case.

For the $\phi_1 U \phi_2$ case, we use a set $\mathbb{Z}$ that begins with the states satisfying ϕ_2 and the fault ratios under which this is the case. Then, we add predecessors of this set that satisfy ϕ_1 and label them with the maximum of the ratio needed to ensure they satisfy ϕ_1 and the least ratio needed to ensure a successor satisfies ϕ_2. This is repeated until a fixed point is reached.

For the $EG\phi$ case, we begin by finding all the states satisfying ϕ and labelling them with the ratio needed for this to be the case. Then, we repeatedly reduce this set to include only those states that can reach a successor within the set and possibly increase the label of the state if all the successors have a higher label than it does. This is repeated until a fixed point is reached.

We now show that the output of λ*−SYNTHESIS* reflects the minimum ratio for which the Fault Tolerance problem would return *false* w.r.t the concrete system and specification given as input. We prove this by means of two steps.

Theorem 1. *Let $\lambda = \lambda$−SYNTHESIS$(\mathcal{S}^f, \phi, n) \neq$ total for a faulty PIIS $\mathcal{S}^f$ and an ACTLK$\backslash$X formula ϕ. Then $\mathcal{S}^f \not\models_n^\lambda \phi$.*

Proof sketch. By definition of λ*−SYNTHESIS* and an adaptation of the proof for the CTL labelling algorithm [Clarke *et al.*, 1999], we have that $\neg\phi$ is satisfied in the submodel of $\mathcal{S}^f((0,n))$ built only from states in which the ratio of agents exhibiting faults is less than or equal to λ. Consequently, we have $\mathcal{S}^f \not\models_n^\lambda \phi$. □

Before proceeding with the next part of the proof, we show some intermediate results. Denote by G the set of global states of $\mathcal{S}^f((0,n))$ and G' the set of global states of $\mathcal{S}^f((n_n, n_f))$. Refer to the agents in $\mathcal{S}^f((0,n))$ by $\mathcal{A}$ and to the agents in $\mathcal{S}^f((n_n, n_f))$ by $\mathcal{A}'$.

Definition 8. *A bijection $\xi : \mathcal{A} \mapsto \mathcal{A}'$ is fault-preserving at some global state $g \in G$ iff every agent $i \in \mathcal{A}$ that has never exhibited a fault in state g is mapped to a non-faulty agent $\xi(i) \in \mathcal{A}'$.*

We define a global state $g \in G$ and a global state $g' \in G'$ to be similar, denoted $g' \approx g$, iff there is some fault-preserving bijection ξ such that for every $i \in \mathcal{A}$ it is the case that $g.i = g'.\xi(i)$, i.e. every agent is in the same local state as the agent it is mapped to.

Definition 9. *A simulation relation $\sqsubset \subseteq G' \times G$ is defined by $g' \sqsubset g$ iff:*

- *$g' \approx g$*
- *For every path π' in $\mathcal{S}^f((n_n, n_f))$ there is a path π in $\mathcal{S}^f((0,n))$ such that $\pi'(i) \sqsubset \pi(i)$ for every $i \geq 0$.*
- *For every g'^1 with $g'.i = g'^1.i$ there is a g^1 with $g.i = g^1.i$ and $g'^1 \sqsubset g^1$.*

Lemma 1. *If $g' \approx g$ then $g' \sqsubset g$.*

Proof sketch. The proof idea is to let each faulty agent i in $\mathcal{S}^f((0,n))$ mimic the agent $\xi(i)$ in $\mathcal{S}^f((n_n, n_f))$. □

We can now proceed to prove the second part of the correctness of the λ*−SYNTHESIS* procedure.

Theorem 2. *Let $\lambda = \lambda$−SYNTHESIS$(\mathcal{S}^f, \phi, n) \neq$ total for a faulty PIIS $\mathcal{S}^f$ and an ACTLK$\backslash$X formula ϕ. Then for all $\lambda' < \lambda$ we have that $\mathcal{S}^f \models_n^{\lambda'} \phi$.*

Proof sketch. Let $n_n, n_f \in \mathbb{Z}$ be such that $n_n + n_f = n$ and $n_f/n = \lambda' < \lambda$. We need to show that $\mathcal{S}^f((n_n, n_f)) \models \phi$.

Let $label(\psi)$ denote the set of minimal labels computed by $\texttt{Lambda}(\mathcal{S}^f((0,n)), \psi)$. We show, by structural induction on ψ, that $g' \sqsubset g \wedge \exists \rho : (g, \rho) \in label(\neg\psi)$ implies either one of the following: $g' \models \neg\psi$, if ψ is a propositional formula; $g' \models \psi$, if ψ is not propositional. Then, we can apply this to the initial states with $\rho = \lambda$ to get our result. The propositional and boolean cases are straightforward. The temporal cases follow from the existentiality of ψ and Lemma 1.

We show the case of $\psi = K_i\psi'$. We have $(g, \rho) \in label(\overline{K}_i\neg\psi')$. We need to show that $g' \models K_i\psi$. Suppose $g' \not\models K_i\psi$. Then, there is a state g'^1 with $g'.i = g'^1.i$ and $g'^1 \models \neg\psi$. By Lemma 1 there is a state g^1 in $\mathcal{S}^f(n)$ with $g'^1 \sqsubset g^1$, $ratio(g^1) = ratio(g'^1) \leq \lambda' < \lambda$ and $g^1 \models \neg\psi$. Therefore $\exists\rho$: $(g^1, \rho) \in label(\neg\psi)$. Since $ratio(g^1) < \lambda$ the latter contradicts that λ is minimum. □

Together, Theorem 1 and Theorem 2 show that Algorithm 1 is a sound decision procedure for the λ-synthesis problem.

4 Parameterised Ratio Synthesis

We now introduce a notion of parameterised fault tolerance similar to that in [Kouvaros and Lomuscio, 2017], but, instead of checking a system with a given ratio of faults against a specification, we here determine the minimum ratio of faults for the specification to be violated.

Definition 10 (Parameterised Fault Tolerance). *Given a faulty PIIS $\mathcal{S}^f$, a ratio $\lambda \in [0,1]$, and an ACTLK$\backslash$X formula ϕ, we say that $\mathcal{S}^f$ is λ-tolerant with respect to ϕ if it is the case that $\mathcal{S}^f \models^\lambda_x \phi$ for all $x \in \mathbb{Z}^+$. We denote this by $\mathcal{S}^f \models^\lambda \phi$.*

We wish to synthesize the least λ that falsifies a specification, which we formalise in the decision problem below.

Definition 11 (Parameterised λ-synthesis). *Given a faulty PIIS $\mathcal{S}^f$ and an ACTLK$\backslash$X formula ϕ, find $\lambda \in [0,1]$ such that $\mathcal{S}^f \not\models^\lambda \phi$ and $\mathcal{S}^f \models^{\lambda'} \phi$ whenever $\lambda' < \lambda$ or return* total *if no such λ exists (i.e. the specification is always satisfied).*

Instead of tacking this problem directly, we consider a discrete version of it.

Definition 12 (Discrete Parameterised λ-synthesis). *Given a faulty PIIS $\mathcal{S}^f$, an ACTLK$\backslash$X formula ϕ and a discretisation step $d \in \mathbb{Z}^+$, find the solution to parameterised λ-synthesis to within $1/d$.*

Notice that a complete decision procedure for this problem cannot exist, as this would give a decision procedure for the parameterised model checking problem, which is known to be undecidable [Apt and Kozen, 1986]. Nonetheless, as for the parameterised model checking problem, it is of interest to explore decidable fragments [Emerson and Namjoshi, 1995; Kouvaros and Lomuscio, 2016]. We do this in the procedure shown in Algorithm 3.

The procedure takes as input the faulty PIIS $\mathcal{S}^f$ and the step d and then calls a cutoff identification procedure [Kouvaros and Lomuscio, 2013a; 2013b; 2015] to identify a cutoff c for the system composed solely of faulty agents, i.e., the PIIS $\langle T^f, e^f, V^f \rangle$. Since cutoffs do not exist in general, by necessity the cited identification procedures are incomplete. In the case a cutoff cannot be identified, then our procedure fails. In the case a cutoff is identified, then our procedure calls the λ*-SYNTHESIS* procedure on each concrete instance up to the maximum of the cutoff and the discretisation step and returns the minimum result of these calls. We aim to prove that, to within $1/d$, this is the least ratio for which ϕ is falsified. As in the previous section, we do this in two steps.

Algorithm 3 Parameterised λ-Synthesis Decision Procedure

```
1: procedure P-λ-SYNTHESIS( $\mathcal{S}^f, \phi, d$ )
2:     Identify a cutoff $c$ for $\langle T^f, e^f, V^f \rangle$ or return fail;
3:     $m \leftarrow$ total;
4:     for $n \leftarrow 1$ to $\max(d, c)$ do
5:         $m \leftarrow \min(m, \lambda\text{-SYNTHESIS}(\mathcal{S}^f, \phi, n))$;
               where we define $\min(\text{total}, x) = x$
6:     end for
7:     return $m$;
8: end procedure
```

Theorem 3. *Let $\lambda = P\text{-}\lambda\text{-}SYNTHESIS(\mathcal{S}^f, \phi, d) \neq$* total *for a faulty PIIS $\mathcal{S}^f$, an ACTLK$\backslash$X formula ϕ and a discretisation step $d \in \mathbb{Z}^+$. Then $\mathcal{S}^f \not\models^\lambda \phi$.*

Proof. Note that for some n it is the case that $S^f \not\models^\lambda_n \phi$ by the correctness of λ*-SYNTHESIS*. The result follows immediately. □

Theorem 4. *Let $\lambda = P\text{-}\lambda\text{-}SYNTHESIS(\mathcal{S}^f, \phi, d) \neq$* total *for a faulty PIIS $\mathcal{S}^f$, an ACTLK$\backslash$X formula ϕ and a discretisation step $d \in \mathbb{Z}^+$. Then for all $\lambda' < \lambda - (1/d)$ we have that $\mathcal{S}^f \models^{\lambda'} \phi$.*

Proof sketch. Let $n_n + n_f = n$ with $n_f/n = \lambda' < \lambda$. We need to show $\mathcal{S}^f((n_n, n_f)) \models \phi$.

Denote by c the cutoff for $\langle T^f, e^f, V^f \rangle$. The cases where $n \leq max(d, c)$ follow directly from the correctness of λ*-SYNTHESIS*. So, it remains to consider when $n > max(d, c)$. Suppose for a contradiction $\mathcal{S}^f((n_n, n_f)) \models \neg\phi$. Then, the initial state of $\mathcal{S}^f((0, n))$ would be labelled with some $\lambda'' \leq \lambda' < \lambda$ by $\texttt{Lambda}(\mathcal{S}^f((0,n)), \neg\phi)$. Now consider the PIIS $S^{\lambda''}$ which is a restriction of $\langle T^f, e^f, V^f \rangle$ to allow at most a ratio of λ'' faulty agents. By the labelling algorithm we have that $S^{\lambda''}(n) \models \neg\phi$. Notice that $max(d, c)$ is also a cutoff for $S^{\lambda''}$ so it follows from this that $S^{\lambda''}(max(d, c)) \models \neg\phi$. But now we have $\mathcal{S}^f \models^{\lambda''}_{max(d,c)} \phi$ for $\lambda'' < \lambda - (1/d)$, contradicting the minimality of the return value for λ*-SYNTHESIS*$(\mathcal{S}^f, \phi, max(d, c))$. □

Together, Theorem 3 and Theorem 4 show that Algorithm 3 is a sound decision procedure for the discrete parameterised λ-synthesis problem.

5 Evaluation

We implemented our algorithms in MCMAS-PFTS, a toolkit constructed from MCMAS-P, a model checker for the verification of PIIS [Kouvaros and Lomuscio, 2013b]. The tool takes

Example	Cutoff	FTR	Synthesis	Iterative
TGC	3	33%	1sec	1sec
Transport	2	50%	16sec	36sec
Alpha	4	33%	986sec	1980sec

Table 1: Experimental results obtained with MCMAS-PFTS.

as input a description of the agent template and the faults to be injected. It then constructs the faulty agent, using the procedures detailed in [Kouvaros and Lomuscio, 2017].

We used MCMAS-PFTS to assess the fault-tolerance of the train-gate-controller, a commonly used benchmark for the verification of MAS, and transport and aggregation protocols in robot swarms. For simplicity, we took the discretisation step to be equal to the cutoff in every case. We are not aware of any other approaches for computing the fault-tolerance ratio of a MAS but to compare the run time of our synthesis approach to a benchmark, we compare it to the run time of the the naïve approach of iteratively checking concrete systems for all possible ratios of faulty to non-faulty agents using the techniques in [Kouvaros and Lomuscio, 2017]. Our results are summarised in Table 1.

Train-Gate-Controller. We modelled the train-gate-controller example from [van der Hoek and Wooldridge, 2002] in which arbitrarily many trains are trying to pass through a tunnel. A controller needs to ensure that only one is present in the tunnel at any given time. We modelled the fault representing that a train might remain stuck in the tunnel when it should have left and checked the property "if a train is in the tunnel, it knows no other train is," expressed in ACTLK$\backslash X$ as

$$\phi_1 \triangleq \forall i, j\colon AG((tunnel, i) \rightarrow K_i \neg (tunnel, j)).$$

We found a cutoff of 3, giving us a fault tolerance ratio of 33%. For this small example, there is no improvement over the naïve approach with both finding the result in 1 second.

Collective transport. We modelled a simplified version of the algorithm described in [Ferrante *et al.*, 2013; Brambilla, 2014]. In this protocol a group of robots are physically connected to an object and need to move it to a target location. However, the robots have limited sensing so may not all perceive the goal and thus need to agree on a direction to move to. Robots that are perceiving the goal will choose a suitable direction and broadcast it to other robots. All the robots, whether they can perceive the goal or not, will calculate the average of the received directions, and collectively move in the same direction.

We made a few assumptions to model the system. In particular, we discretised time, space and direction of movement and assumed that the robots are working on a finite 5×5 grid. The grid is allowed to wrap around, so $(5, 1)$ is left of $(1, 1)$ and $(1, 5)$ is above it. We also assumed that each robot only receives broadcasts from 3 of its neighbours. We initialised the robots at coordinates either $(1, 1)$ or $(1, 2)$ or $(2, 1)$ or $(2, 2)$ and their goal located at $(3, 5)$. We modelled a fault consisting of a robot choosing to move in the wrong direction. We checked the property "all non-faulty robots are always moving towards the target," expressed by the formula

$$\phi_2 \triangleq \forall i\colon AG(to\_target, i).$$

We found a cutoff of 2, giving us a fault tolerance ratio of 50% in 16 seconds, compared to 36 seconds using the iterative approach. Hence, the procedure here introduced was an order of magnitude faster than the naïve approach.

Alpha algorithm. Finally, we assessed the fault-tolerance of the Alpha swarm aggregation algorithm [Winfield *et al.*, 2008], a protocol used to aggregate robotic swarms in the areas of operation. Define a robot to be in another robot's neighbourhood if the position of the former is in the range of the latter's sensor. Each robot keeps track of the number of its neighbours. A robot is said to be *connected* if its neighbourhood is composed of at least α robots, for a threshold α. The behaviour of each of the robots is characterised by their connectivity status and by whether they are in *forward motion mode* or in *coherence motion mode*: if a robot is in forward mode and connected, then it moves forward; if it is in forward mode, but not connected, then it performs a 180° turn and changes its motion mode to coherence; if it is in coherence mode, but not connected, then it moves forward; finally, if it is in coherence mode and connected, then it performs a random 90° turn and changes its motion mode to forward.

We adopted a number of assumptions commonly made to analyse the algorithm [Kouvaros and Lomuscio, 2017; Dixon *et al.*, 2012]. In particular, we fixed a 5×5 arena (once again wrapping around), assumed a communication range of 1, and let $\alpha = 2$. Initially the robots are connected, in forward mode, and collectively have every possible direction of movement. Formally, we adopted the PIIS proposed in [Kouvaros and Lomuscio, 2015]. We injected *direction faults* which inhibit a robot from changing direction when it should have as a result of, for example, a mechanical failure. We are here interested in analysing the swarm's *connectedness property* [Dixon *et al.*, 2012] "every non-faulty robot knows that it will be infinitely often connected," expressed in ACTLK$\backslash X$ as

$$\phi_3 \triangleq \forall i\colon K_i GF(connected, i).$$

We found a cutoff of 4, giving us a fault tolerance ratio of 33% in 986 seconds, compared to 1980 seconds obtained via the iterative approach. As in the previous example, the procedure was an order of magnitude faster.

6 Conclusions

We have presented a method for computing the ratio of faulty agents that a protocol can tolerate whilst still satisfying a specification. Calculating the ratio is important when deploying systems as potential failure risks can then be assessed and remedial action can be taken. Alternatively, if a protocol is found to be very robust to errors, more faults can be tolerated at runtime, thereby possibly generating savings.

We are not aware of other approaches for addressing the same problem. When performing the evaluation we compared the approach to an iterative procedure of our construction built from [Kouvaros and Lomuscio, 2017]. We found the present approach to be faster by an order of magnitude on significant examples.

In future work we intend to apply this methodology to study resilience properties in further swarm algorithms.

References

[Aminof *et al.*, 2018] Benjamin Aminof, Sasha Rubin, Ilina Stoilkovska, Josef Widder, and Florian Zuleger. Parameterized model checking of synchronous distributed algorithms by abstraction. In *Proceedings of VMCAI18*, pages 1–24. Springer International Publishing, 2018.

[Apt and Kozen, 1986] Krzysztof R. Apt and Dexter Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, 22(6):307–309, 1986.

[Barnat *et al.*, 2012] Jiri Barnat, Lubos Brim, Adam Krejci, Adam Streck, David Safranek, Martin Vejnar, and Tomas Vejpustek. On parameter synthesis by parallel model checking. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(3):693–705, 2012.

[Beneš *et al.*, 2016] Nikola Beneš, Luboš Brim, Martin Demko, Samuel Pastva, and David Šafránek. Parallel SMT-based parameter synthesis with application to piecewise multi-affine systems. In *Proceedings of ATVA16*, volume 9938 of *LNCS*, pages 192–208. Springer, 2016.

[Bloem *et al.*, 2015] Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Veith Veith, and Josef Widder. *Decidability of Parameterized Verification*. Morgan and Claypool Publishers, 2015.

[Bozzano and Villafiorita, 2007] Marco Bozzano and Adolfo Villafiorita. The FSAP/NuSMV-SA safety analysis platform. *Software Tools for Technology Transfer*, 9(1):5–24, 2007.

[Brambilla, 2014] Manuele Brambilla. *Formal methods for the design and analysis of robot swarms*. PhD thesis, Ecole Polytechnique de Bruxelles, 2014.

[Clarke *et al.*, 1999] Edmund M. Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[Dixon *et al.*, 2012] Clare Dixon, Alan FT Winfield, Michael Fisher, and Chengxiu Zeng. Towards temporal verification of swarm robotic systems. *Robotics and Autonomous Systems*, 60(11):1429–1441, 2012.

[Emerson and Namjoshi, 1995] E. Allen Emerson and Kedar S. Namjoshi. Reasoning about rings. In *Proceedings of POPL95*, pages 85–94. Pearson Education, 1995.

[Ezekiel and Lomuscio, 2009] Jonathan Ezekiel and Alessio Lomuscio. Combining fault injection and model checking to verify fault tolerance in multi-agent systems. In *Proceedings of AAMAS09*, pages 113–120. IFAAMAS Press, 2009.

[Ezekiel and Lomuscio, 2017] Jonathan Ezekiel and Alessio Lomuscio. Combining fault injection and model checking to verify fault tolerance, recoverability, and diagnosability in multi-agent systems. *Information and Computation*, 254(2):167–194, 2017.

[Ezekiel *et al.*, 2011] Jonathan Ezekiel, Alessio Lomuscio, Levente Molnar, and Sandor Veres. Verifying fault tolerance and self-diagnosability of an autonomous underwater vehicle. In *Proceedings of IJCAI11*, pages 1659–1664. AAAI Press, 2011.

[Ferrante *et al.*, 2013] Eliseo Ferrante, Manuele Brambilla, Mauro Birattari, and Marco Dorigo. *Socially-Mediated Negotiation for Obstacle Avoidance in Collective Transport*, volume 83 of *STAR*, pages 571–583. Springer Berlin Heidelberg, 2013.

[van der Hoek and Wooldridge, 2002] Wiebe van der Hoek and Michael Wooldridge. Tractable multiagent planning for epistemic goals. In *Proceedings of AAMAS02*, pages 1167–1174. ACM Press, 2002.

[John *et al.*, 2013] Annu John, Igor Konnov, Ulrich Schmid, Helmut Veith, and Josef Widder. Parameterized model checking of fault-tolerant distributed algorithms by abstraction. In *Proceedings of FMCAD13*, pages 201–209. IEEE, 2013.

[Kouvaros and Lomuscio, 2013a] Panagiotis Kouvaros and Alessio Lomuscio. Automatic verification of parameterised interleaved multi-agent systems. In *Proceedings AAMAS13*, pages 861–868. IFAAMAS, 2013.

[Kouvaros and Lomuscio, 2013b] Panagiotis Kouvaros and Alessio Lomuscio. A cutoff technique for the verification of parameterised interpreted systems with parameterised environments. In *Proceedings of IJCAI13*, pages 2013–2019. AAAI Press, 2013.

[Kouvaros and Lomuscio, 2015] Panagiotis Kouvaros and Alessio Lomuscio. Verifying emergent properties of swarms. In *Proceedings of IJCAI15*, pages 1083–1089. AAAI Press, 2015.

[Kouvaros and Lomuscio, 2016] Panagiotis Kouvaros and Alessio Lomuscio. Parameterised verification for multi-agent systems. *Artificial Intelligence*, 234:152–189, 2016.

[Kouvaros and Lomuscio, 2017] Panagiotis Kouvaros and Alessio Lomuscio. Verifying fault-tolerance in parameterised multi-agent systems. In *Proceedings of IJCAI17*, pages 288–294. AAAI Press, 2017.

[Lomuscio *et al.*, 2010] Alessio Lomuscio, Wojciech Penczek, and Hongyang Qu. Partial order reduction for model checking interleaved multi-agent systems. *Fundamenta Informaticae*, 101(1–2):71–90, 2010.

[Winfield *et al.*, 2005] Alan FT Winfield, Chrostopher J. Harper, and Julien Nembrini. Towards dependable swarms and a new discipline of swarm engineering. In *Proceedings of SAB2004*, volume 3342 of *LNCS*, pages 126–142. Springer, 2005.

[Winfield *et al.*, 2008] Alan FT Winfield, Wenguo Liu, Julien Nembrini, and Alcherio Martinoli. Modelling a wireless connected swarm of mobile robots. *Swarm Intelligence*, 2(2-4):241–266, 2008.

[Zhang *et al.*, 2009] Yingqian Zhang, Efrat Manisterski, Sarit Kraus, VS Subrahmanian, and David Peleg. Computing the fault tolerance of multi-agent deployment. *Artificial Intelligence*, 173(3):437–465, 2009.