Reconfigurability in Reactive Multiagent Systems

Xiaowei Huang[1,2], **Qingliang Chen**[1], **Jie Meng**[3], **Kaile Su**[1,4]
[1]Department of Computer Science, Jinan University, China
[2]Department of Computer Science, University of Oxford, United Kingdom
[3]Department of Marketing and Management, Macquarie University, Australia
[4]Institute for Integrated and Intelligent Systems, Griffith University, Brisbane, Australia

Abstract

Reactive agents are suitable for representing physical resources in manufacturing control systems. An important challenge of agent-based manufacturing control systems is to develop formal and structured approaches to support their specification and verification. This paper proposes a logic-based approach, by generalising that of model checking multiagent systems, for the reconfigurability of reactive multiagent systems. Two reconfigurability scenarios are studied, for the resulting system being a monolithic system or an individual module, and their computational complexity results are given.

1 Introduction

Industrie 4.0 [Ind, 2011], the German vision for the future of manufacturing, recognises modular structured smart factories as one of the key ingredients. Smart factories use information and communications technologies to digitise their processes and reap huge benefits in the form of improved quality, lower costs, and increased efficiency [Zaske, 2015]. Agent-based systems have been applied to solve complex manufacturing system problems including manufacturing planning, scheduling and control, and supply chain management [Tang and Wong, 2005]. In particular, agents can represent physical resources such as machine tools, robots, auto-guided vehicles, etc [Paulo Leitão, 2009]. However, the current development of such manufacturing control systems starts from simple graphical specifications and relies on the code developers to take care of the implementation details. This development approach not only is inefficient (in terms of the time and costs of the development) but also raises concerns about the reliability, performance, reusability, and reconfigurability of the resulting solution. Therefore, as pointed out in [Marik and McFarlane, 2005; Paulo Leitão, 2009], an important challenge for agent-based manufacturing control systems is to develop formal and structured approaches to support their specification and verification.

In this paper, we introduce a logic-based approach for the specification and verification of manufacturing control systems (MCS), by generalising the existing approach of model checking multiagent systems. An MCS is a multiagent system, i.e., there are a set of agents interacting and communicating in an environment. In an MCS, each agent has limited, i.e., incomplete, information about the system by e.g., sensing devices. For instance, robotic arms fetching the same raw materials from a distributor robot are not supposed to observe the status (i.e., ready to fetch or not) of each other. The other characteristics of the agents in an MCS is reactivity, i.e., they behave in the way of reacting to the stimulus from the outside world [Tang and Wong, 2005]. Therefore, no memory is needed. Both incomplete information and reactivity of agents are not new in the area of model checking multiagent systems, with theoretical results [Fagin *et al.*, 1995; Alur *et al.*, 2002] and prototype verification softwares [Gammie and van der Meyden, 2004; Lomuscio *et al.*, 2009].

The paper addresses a new challenge in the development of MCSs and Industrie 4.0 [Nikolaus, 2014], i.e., reconfigurable robots, which are robots whose modules can be connected in different ways to form different robots in terms of size, shape, or function [Chen *et al.*, 2006; Kasper Støy *et al.*, 2010; Dennis *et al.*, 2014]. Reconfigurable manufacturing systems can quickly adjust their production capacity and functionality in response to sudden market changes or intrinsic system change [Koren *et al.*, 2003]. In particular, the paper focuses on the decision problem for reconfigurable robots, called reconfigurability in the paper. Informally, the reconfigurability is, given a set of reactive robots, to determine whether they can form a composite robot with designated functionalities.

Two different development scenarios are considered. The first is the scenario in which the resulting system is a monolithic system[1]. The reconfigurability is to determine the existence of agents' reactive behaviour to satisfy their local functionalities and the system's global functionalities. Following the success of model checking [Clarke *et al.*, 1999], a logic language is employed to describe the functionalities, by extending the temporal epistemic logic CTLK [Fagin *et al.*, 1995] with the expressiveness to refer to agents' reactive behaviour directly in the formula. We show that the complexity of the problem is NP-complete.

The second is the scenario where the resulting system is an individual module of a bigger system. The resulting system has a set of nondeterministic transitions, called environment transitions in the paper, which are to accept interactions from

[1]To differentiate component robots with the resulting robot, we write agents for component robots and system for the resulting robot.

other modules of the bigger system. The reconfigurability is to determine the existence of agents' reactive behaviour in satisfying the required functionalities, under all possible interactions. We show that the complexity of the problem is EXPTIME-complete.

2 Model Checking Multiagent Systems

A multiagent system consists of a set $Agt = \{1, ..., n\}$ of agents running in an environment [Fagin *et al.*, 1995]. At each time, each agent is in some *local state*. The environment, regarded as a special agent e, is in some *environment state*, which keeps track of everything relevant to the system but not recorded in agents' local states. A global state s is a tuple $(s_e, s_1, ..., s_n)$ consisting of an environment state s_e and a local state s_i for each agent i. Let L_i, for $i \in Agt$, be a set of agent i's local states, and L_e be a set of environment states.

Each agent i has a nonempty set $Acts_i$ of local actions. Let $Acts = \Pi_{i \in Agt} Acts_i$ be a set of joint actions. The environment runs a protocol $Prot_e = (L_e, t_e, T_e, \{P_i\}_{i \in Agt})$ where $t_e \in L_e$ is an initial state[2], $T_e : L_e \times Acts \to \mathcal{P}(L_e) \setminus \{\emptyset\}$ is a labeled transition function updating environment states according to the current state and agents' joint action, and $P_i : L_e \to O_i$ provides the perception/observation of agent i on the environment states. Each agent i runs a protocol $Prot_i = (L_i, Acts_i, LAct_i, t_i, T_i)$, where function $LAct_i : L_i \times O_i \to \mathcal{P}(Acts_i) \setminus \{\emptyset\}$ defines for each local state and observation a nonempty set of actions that can be taken by agent i, $t_i \in L_i$ is an initial local state, and $T_i : L_i \times O_i \times Acts_i \to \mathcal{P}(L_i) \setminus \{\emptyset\}$ is a labeled transition function. Let V be a finite set of boolean variables. Formally[3], a multiagent system is a tuple $M = (S, t, T, \{\sim_i\}_{i \in Agt}, \pi, \mathcal{F})$ where

1. $S \subseteq L_e \times \Pi_{i \in Agt} L_i$ is a set of global states,
2. $t = (t_e, t_1, ..., t_n)$ is an initial global state,
3. $T \subseteq S \times S$ is a transition relation such that for all states $s = (l_e, l_1, ..., l_n)$ and $s' = (l'_e, l'_1, ..., l'_n)$, we have $(s, s') \in T$ iff there exist local actions $a_1, ..., a_n$ such that for all $i \in Agt$, there are $a_i \in LAct_i((l_i, P_i(l_e)))$ and $(l_i, P_i(l_e), a_i, l'_i) \in T_i$, and for the environment, there is $(l_e, (a_1, ..., a_n), l'_e) \in T_e$,
4. $\sim_i \subseteq S \times S$ is an indistinguishable relation of agent i such that for all states $s = (l_e, l_1, ..., l_n)$ and $s' = (l'_e, l'_1, ..., l'_n)$, we have $(s, s') \in \sim_i$ iff $l_i = l'_i$ and $P_i(l_e) = P_i(l'_e)$,
5. $\pi : S \to \mathcal{P}(V)$ assigns each global state with a set of boolean variables in V, and
6. $\mathcal{F} \subseteq S$ is a Büchi acceptance condition.

A *path* in M is a finite or infinite sequence $s_0 s_1 s_2 ...$ such that $(s_i, s_{i+1}) \in T$ for all $i \geq 0$. An infinite path $s_0 s_1 s_2 ...$ is *fair* with respect to $\mathcal{F}$ if there are infinitely many indices j for which $s_j \in \mathcal{F}$. Let $rch(M)$ be the set of *fair reachable* states of M, i.e., the set of states s_n (for some n) such that there exists a fair path $s_0 s_1 ... s_n s_{n+1} ...$ with $s_0 = t$ the initial state.

The language CTLK(V, Agt) has the syntax:

$$\phi ::= v \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid EX\phi \mid E(\phi_1 U \phi_2) \mid EG\phi \mid K_i\phi$$

where $v \in V$ and $i \in Agt$. Other operators are defined as usual, e.g., $AF\phi = \neg EG\neg\phi$ and $AG\phi = \neg E(true\, U \neg\phi)$. The semantics of the language is given by a satisfaction relation $M, s \models \phi$, where $s \in rch(M)$ is a fair reachable state. This relation is defined inductively as follows:

1. $M, s \models v$ if $v \in \pi(s)$,
2. $M, s \models \neg\phi$ if not $M, s \models \phi$,
3. $M, s \models \phi_1 \vee \phi_2$ if $M, s \models \phi_1$ or $M, s \models \phi_2$,
4. $M, s \models EX\phi$ if there exists $s' \in rch(M)$ such that $(s, s') \in T$ and $M, s' \models \phi$,
5. $M, s \models E(\phi_1 U \phi_2)$ if there exists a fair path $s_0 s_1 \ldots s_n \ldots$ such that $s_0 = s$, $M, s_k \models \phi_1$ for $0 \leq k \leq n-1$ and $M, s_n \models \phi_2$,
6. $M, s \models EG\phi$ if for there exists a fair path $s_0 s_1 \ldots$ such that $s_0 = s$ and $M, s_k \models \phi$ for all $k \geq 0$,
7. $M, s \models K_i\phi$ if for all $s' \in rch(M)$ with $(s, s') \in \sim_i$ we have $M, s' \models \phi$.

Given a multiagent system M and a CTLK(V, Agt) formula ϕ, the model checking problem, written as $M \models \phi$, is to decide whether $M, t \models \phi$. It is shown [Clarke *et al.*, 1999] that, if we measure the problem with the number $|S|$ of states and the length $|\phi|$ of formula, then it is PTIME-complete.

Example 1 *We consider a typical MCS in which a (distributor) robot distributes semi-finished products to a set of (receiver) robots. Robots communicate with each other by wireless signals and self-organise themselves to ensure the success of this distribution task. The factory floor has limited space, so that at every time only one receiver can be allowed to access the distributor for the product.*

In the system, each receiver i may be in one of the three states $\{w_i, t_i, c_i\}$, which represents that it is waiting (i.e., it is doing other things irrelevant to the product), trying (i.e., it has decided to access the distributor as its next job), or collecting (i.e., it is accessing the distributor), respectively. Two signals r_i and d_i are designed for each receiver i to communicate with the distributor. The signal $r_i = 1$ represents that the receiver i is notifying the distributor with its intention to get the product, and the signal $d_i = 1$ represents that the distributor is notifying the receiver i that it is allowed to access. The distributor, modelled by the environment, may be in one of the two states $\{f, e\}$, representing that it is full (i.e., there is a receiver allowed to access) or empty. We let $L_i = \{w_i, t_i, c_i\}$ and $L_e = \{f, e\} \times \Pi_{i \in Agt}\{r_i, d_i\}$.

We define transition relations T_i and T_e by describing the effects of actions. The receiver i has a set $\{a_{i,\top}, a_{i,t}, a_{i,c}, a_{i,w}\}$ of actions. At state w_i, it can take either $a_{i,\top}$ to stay still or $a_{i,t}$ to start trying. The effects of $a_{i,t}$ include turning the state into t_i and letting the signal $r_i = 1$. At state t_i, it can take either $a_{i,\top}$ to stay still or $a_{i,c}$ to start accessing the materials.

[2]For both the environment and the agents, we assume a single initial state. This is to ease the technicalities that will be employed in the complexity proofs. However, the results of this paper can be generalised to the case where there are a set of initial states.

[3]Although simple, this formalisation is sufficiently expressive in modelling MCSs by engaging certain levels of abstractions. For example, the sensing results of robots can be modelled by agents' observations, and the communication and interactions between robots can be modelled by agents' joint actions.

The effect of $a_{i,c}$ is turning the state into c_i. At state c_i, it can take either $a_{i,\top}$ to stay still or $a_{i,w}$ to leave the distributor. The effects of $a_{i,w}$ include turning the state into w_i and letting the signal $d_i = 0$. For the distributor, at state e, it nondeterministically chooses one of the receivers i that has sent a request, changing its own state into f, and letting $d_i = 1$ and $r_i = 0$. The distributor does nothing if it is at state f. Whenever a receiver takes action $a_{i,w}$, the distributor will update its state from f to e.

For the observation function P_i, we assume that receiver i can observe the signals r_i and d_i. Initially, we have w_i as the state of all receivers i, e as the state of the distributor, and $r_i = 0$ and $d_i = 0$ as the states of the signals. There is no fairness condition required for the system and the labelling function is defined by directly referring to the states of the receivers, the distributor, or the signals.

For such a system, the following specification formulas describe a set of criteria for the success of the distribution task:

- $\phi_1 \equiv \bigwedge_{i\in Agt} AG(t_i \Rightarrow AFc_i)$ *expresses that once a receiver starts trying, it can eventually get the product,*
- $\phi_2 \equiv \bigwedge_{i\in Agt} AGAF\neg c_i$ *expresses that none of the receivers can access the distributor without exiting, and*
- $\phi_3 \equiv AG\bigwedge_{i\in Agt}\bigwedge_{j\in Agt}(i \neq j \Rightarrow \neg c_i \vee \neg c_j)$ *expresses that there will not be the case in which two receivers are allowed to access the distributor at the same time.*
- $\phi_4 \equiv AG\bigwedge_{i\in Agt}(d_i \Rightarrow K_i\bigwedge_{j\in Agt}(j \neq i \Rightarrow \neg d_i))$ *expresses that once a receiver is given the access, it knows that none of the other receivers has been given the access.*

However, it can be checked that none of the above specification formulas can be satisfiable on the described system.

3 Reconfigurability vs. Model Checking MAS

The example in the previous section shows an insufficiency of the approaches of model checking multiagent systems in handling interesting examples in smart factories. The techniques of model checking multiagent systems assume a closed system in which the behaviour of the agents and the environment is completely specified with their respective protocols. Therefore, the system in Example 1 is regarded as *under-specified* because only the available actions of the agents and their effects are described. To have a meaningful use of the model checking techniques, more details are needed to complete the protocols, including the following two aspects:

A1. the conditions (or guards, etc) for the agents to take local actions, that is, the agents can take the actions only if their respective conditions are satisfied; for example, in Example 1, conditions may be needed for action $a_{i,c}$, and different agents may have different conditions.

A2. the interaction of the system, as a module of a bigger system, with other modules; for example, in Example 1, a resolution of the nondeterminism of the distributor on choosing a specific receiver to satisfy its request may exist if there is an interaction with other modules.

While the above discussion suggests that we need to complete the under-specified protocols, it is arguable that, for the reconfiguration of reactive multiagent systems, *the information about the available actions and their effects has been sufficient*. These are exact information that we can expect for the component robots from their factory specifications. Based on these information, a reconfiguration is to find a way for the robots to be connected and communicated so that the resulting system can satisfy some objectives.

An important implication from the above arguments is that, for reconfiguration, instead of regarding the protocols as describing the actual behaviour, it is more reasonable to assume that the protocols describe the *allowed or legal behaviour* according to the robots' factory specifications. Each reconfiguration can then be concretised as a set of restrictions on the legal behaviour. For the studying of complexity, we consider a decision-problem called *reconfigurability*, which can be generally described as follows: given a set of reactive robots, it is to determine the existence of restrictions on their legal behaviour with respect to the objectives.

Formal definition of reconfigurability, to be given later, is scenario dependent. Here we explain the restrictions which will be imposed on the protocols. We utilise an intuition that the restrictions can be regarded as the inputs from the outside world and there are agents and environment behind their respective protocols to conduct such inputs. At each time, an input to a protocol is a (nonempty) subset of the allowed behaviour. In the following two sections, we will discuss two variants of the reconfigurability based on different scenarios.

4 Reconfigurability of A Monolithic System

The first variant assumes that agents have inputs via config-settings (to be defined below) and the environment does not have additional input. This is to model the scenario in which we are working with a monolithic system, so that we need only take care of the organisation of agents' behaviour and assume that the interactions with other systems have been completely specified in the environment protocol.

An agent's inputs are defined with a set of config-settings, each of which defines whether some action may be selected to take based on its currently available local information, i.e., agent's local state and current observation[4]. Formally, for $i \in Agt$ and $a \in Acts_i$, a config-setting $x_{i,a} : L_i \times O_i \to \{0, 1\}$ is a characteristic function over $L_i \times O_i$. Given $x_{i,a}$, the function $LAct_i$ is lifted into $LAct_i[x_{i,a}] : L_i \times O_i \to \mathcal{P}(Acts_i)$ such that, $b \in LAct_i[x_{i,a}]((s_i, o_i))$ for some state $s_i \in L_i$ and observation $o_i \in O_i$ if one of the following conditions holds:

1. $a = b$, $b \in LAct_i((s_i, o_i))$ and $x_{i,a}((s_i, o_i)) = 1$.
2. $a \neq b$ and $b \in LAct_i((s_i, o_i))$.

Intuitively, action a remains legal on state s_i and observation o_i if (s_i, o_i) is permitted by $x_{i,a}$. These can be easily extended to work with a set of local actions $A \subseteq \bigcup_{i\in Agt} Acts_i$ and a set of config-settings $X_A = \{x_{i,a} \mid i \in Agt, a \in A \cap Acts_i\}$. The function $LAct_i$ is lifted into $LAct_i[X_A]$ such that $LAct_i[X_A] = (LAct_i[x])[X_A \setminus \{x\}]$ for any $x \in X_A \neq \emptyset$ and $LAct_i[\emptyset] = LAct_i$. Furthermore, we define multiagent system $M[X_A]$ by lifting functions $LAct_i$ for $i \in Agt$ according to X_A. Note that, for a

[4]This definition is in line with the capability of reactive agents, which works in the stimulus-response way.

specific local information (s_i, o_i), there can be more than one actions a such that $x_{i,a}(s_i, o_i) = 1$.

Given a multiagent system M, a set $A \subseteq \bigcup_{i \in Agt} Acts_i$ of local actions, and a CTLK(V, Agt) formula ϕ, the reconfigurability problem, written as $M, A \models \phi$, is to decide whether there exist config-settings $X_A = \{x_{i,a} \mid a \in A \cap Acts_i, i \in Agt\}$ such that $M[X_A] \models \phi$. The complexity of the problem is measured over the number $|S|$ of states, the number $|\bigcup_{i \in Agt} Acts_i|$ of local actions, and the length $|\phi|$ of formula. We have the following conclusion.

Theorem 1 *The reconfigurability with config-settings is NP-complete for CTLK(V, Agt).*

We have a nondeterministic algorithm that runs in polynomial time. For each action $a \in A$ such that $a \in Acts_i$, the algorithm guesses a set of local states for $x_{i,a}$. The system M is then updated into $M[X_A]$, on which a model checking procedure is held for the formula ϕ. Both the updating of system and the model checking can be done in polynomial time.

We note that, the problem has PTIME reductions to other problems that can be solved by SAT solvers or ASP solvers.

Example 2 *(Continue with Example 1) We let $A = \{a_{i,c} \mid i \in Agt\}$ be the set of local actions, and $\phi \equiv \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4$ be a CTLK(V, Agt) formula. Then with expression $M, A \models \phi$, we can check the existence of conditions for agents to take action $a_{i,c}$, such that after imposing such conditions on agents' protocols, the formula ϕ is satisfiable. It can be verified that, this expression holds for systems with 2 agents.*

The above example shows the advantage of reconfiguration with config-settings over model checking in that, it can work directly with under-specified agents' protocols and automatically verify the formulas without specifying the details of the aspect A1 (given in Section 3) for agents' protocols.

In the following, we extend the language CTLK(V, Agt) into CTLK$(V, Agt, \bigcup_{i \in Agt} Acts_i)$, which has the syntax:

$$\phi ::= \; x_{i,a} \mid v \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid EX\phi \mid E(\phi_1 U \phi_2) \mid EG\phi \mid K_i\phi$$

where $v \in V$, $i \in Agt$, and $a \in Acts_i$. The satisfaction relation of the new construct $x_{i,a}$ is defined as follows.

1. $M, s \models x_{i,a}$ if $x_{i,a}((l_i, P_i(l_e))) = 1$ for $s = (l_e, l_1, ..., l_n)$

As will explain in Example 3, $x_{i,a}$, representing a set of states as that of atomic propositions, can be regarded as an atomic proposition of the resulting system (i.e., system imposed with reconfiguration) that are relevant to the actions. The reconfigurability problem is defined the same as before. The following conclusion shows that the additional expressiveness from the new construct comes for free.

Theorem 2 *The reconfigurability with config-settings is NP-complete for CTLK$(V, Agt, \bigcup_{i \in Agt} Acts_i)$.*

For the algorithm, we note that checking $M[X_A], s \models x_{i,a}$ can be done in linear time. Now we show the usefulness of the additional expressiveness.

Example 3 *(Continue with Example 2) Besides those specification formulas in Example 1, we may be interested in*

- $\phi_5 \equiv AG(x_{i,a_{i,c}} \Leftrightarrow K_i AX \bigwedge_{j \in Agt}(j \neq i \Rightarrow \neg c_j))$*, which expresses that whenever receiver i is permitted to take action $a_{i,c}$, it knows that there is no collision with other receivers in accessing the distributor, and vice versa.*
- $\phi_6 \equiv AG(x_{i,a_{i,c}} \Rightarrow AXc_i)$*, which expresses that whenever receiver i is permitted to take action $a_{i,c}$, it will be accessing the distributor in the next state.*
- $\phi_7 \equiv AG(x_{i,a_{i,c}} \Rightarrow AF\neg x_{i,a_{i,c}})$*, which expresses that receiver i cannot always be accessing the distributor.*

In ϕ_5, ϕ_6, ϕ_7, we use the construct $x_{i,a_{i,c}}$ to directly refer to the behaviour of the receiver i in the resulting system. The benefit of writing such a construct $x_{i,a}$ directly in a formula is that, while there may exist many possible reconfigurations, we can identify some of them that we are interested in by requiring the resulting system to satisfy formulas with constructs $x_{i,a_{i,c}}$. Recall that, $x_{i,a_{i,c}}$ is a characteristic function over $L_i \times O_i$, which means that it can be seen as a representation of a set of states on which action $a_{i,c}$ is allowed to be taken by agent i. For example, with ϕ_6, the resulting system should satisfy the following: on every state where it is allowed to take action $a_{i,c}$, agent i will take that action. Note that, this property cannot be expressed with usual CTLK(V, Agt) formulas, in particular, it is not the same as the formula $AG(t_i \Rightarrow AXc_i)$.

5 Reconfigurability of A Module

In the first variant, the environment's behaviour is completely determined by its protocol, that is, given an environment state l_e and a joint action a, the set of possible next environment states is $N(l_e, a) = \{l'_e \mid (l_e, a, l'_e) \in T_e\}$. For the second reconfigurability variant, we consider a more intriguing problem of allowing not only the inputs from agents but also the inputs from the environment. This is essential when considering a manufacturing system as an individual module of a larger system, which is typical for the vision of smart factories and Industrie 4.0 that tends to consider the connection of different factories and supply chains.

Such scenario suggests that, 1) the reconfiguration is for the agents' behaviour in the system, and 2) agents (and the system) do not have prior knowledge about how the interactions with other systems may occur after reconfiguration. The latter suggests that, to ensure that the resulting system can satisfy objectives, the reconfiguration needs to consider all possible interactions from the environment (Recall that we imagine there is an agent behind the environment protocol to conduct the inputs). Further, the definition of "all possible interactions from the environment" can be concretised by stating that the environment has *complete information* over the system state and *perfect recall memory* over the history. It is a maximal assumption over the environment's capability, but nevertheless reasonable for the reasons given above.

Now we formalise the above intuitions. The environment behaves based on the environment protocol, i.e., given an environment state l_e and a joint action a, the set of possible next environment states can be any nonempty subset of the set $N(l_e, a)$. Formally, the inputs of the environment can be defined by a strategy σ as follows: given any finite path $\rho = ts_1...s_k$ with t the initial state and $s_k = (l_e, l_1, ..., l_n)$ and

any joint action $a = (a_1, ..., a_n)$ such that $a_i \in LAct_i((l_i, P_i(l_e)))$ for all $i \in Agt$, we have that $\sigma(\rho, a) \subseteq N(l_e, a)$ and $\sigma(\rho, a) \neq \emptyset$. Let $\Sigma(M)$ be the set of all environment strategies in a multiagent system M.

Given a strategy σ of the environment, a path $s_0 s_1 s_2 ...$ is consistent with σ if for all $i \geq 0$ with $s_i = (l_e, l_1, ..., l_n)$ and $s_{i+1} = (l'_e, l'_1, ..., l'_n)$, we have that $(s_i, s_{i+1}) \in T$ and there exists a joint action $a = (a_1, ..., a_n)$ such that $l'_e \in \sigma(s_0 ... s_i, a)$ and $(l_i, P_i(l_e), a_i, l'_i) \in T_i$ for all $i \in Agt$. Now we can define the semantics of a formula ϕ over a multiagent system M and an environment strategy σ by a satisfaction relation $M, \sigma, s \models \phi$ where $s \in rch(M)$. The definition basically follows the same pattern as $M, s \models \phi$, except for the following points. When interpreting temporal formulas, paths need to be consistent with σ. E.g.,

1. $M, \sigma, s \models EG\phi$ if there exists a fair path $s_0 s_1 \ldots$ consistent with σ, such that $s_0 = s$ and $M, \sigma, s_k \models \phi$ for all $k \geq 0$,

Let $rch(M, \sigma)$ be a set of states that are fair and reachable in the system M by paths consistent with σ. The semantics of knowledge operator assumes that agents do not have prior knowledge over the environment's strategy. That is,

1. $M, \sigma, s \models K_i\phi$ if for all $s' \in rch(M)$ and all strategies $\sigma' \in \Sigma(M)$ such that $(s, s') \in \sim_i$ and $s' \in rch(M, \sigma')$, we have $M, \sigma', s' \models \phi$.

Considering config-settings under all possible environment strategies, we define the following reconfigurability variant. Given a multiagent system M, a set $A \subseteq \bigcup_{i \in Agt} Acts_i$ of local actions, and a CTLK$(V, Agt, \bigcup_{i \in Agt} Acts_i)$ formula ϕ, the reconfigurability problem, written as $M, A \models_{env} \phi$, is to decide whether there exist config-settings $X_A = \{x_{i,a} \mid a \in A \cap Acts_i, i \in Agt\}$ such that for all strategies $\sigma \in \Sigma(M[X_A])$, we have that $M[X_A], \sigma, t \models \phi$. Note that, according to this definition, the environment has prior knowledge over agents' config-settings. This is in line with the maximal assumption over environment's capability. The complexity of the problem is measured over the number $|S|$ of states, the number $|\bigcup_{i \in Agt} Acts_i|$ of local actions, and the length $|\phi|$ of formula.

As expected, the complexity of reconfigurability with environment strategy is higher than without environment strategy.

Theorem 3 *The reconfigurability with config-settings and environment strategy is EXPTIME-complete for both CTLK(V, Agt) and CTLK$(V, Agt, \bigcup_{i \in Agt} Acts_i)$.*

Proof: (Sketch) Let $A = \{a_1, ..., a_k\}$ be a set of local actions such that for $1 \leq j \leq k$, $a_j \in Acts_{i_j}$ for some agent i_j. Given M, we construct $M' = (S', t', T', \{\sim'_i\}_{i \in Agt}, \pi', \mathcal{F})$ such that

1. $S' = (L_e \times L_1 \times ... \times L_n \times \Pi_{j=1}^{k} \mathcal{P}(L_{i_j} \times O_{i_j})) \cup \{t'\}$,
2. the transition relation T' is defined as follows.
   (a) $(t', (t_e, t_1, ..., t_n, y_1, ..., y_k)) \in T'$ if for all $1 \leq j \leq k$, there is $y_j \in \mathcal{P}(L_{i_j} \times O_{i_j})$.
   (b) $((l_e, l_1, ..., l_n, y_1, ..., y_k), (l'_e, l'_1, ..., l'_n, y_1, ..., y_k)) \in T'$ if there exist local actions $a'_1, ..., a'_n$ such that $(l_e, (a'_1, ..., a'_n), l'_e) \in T_e$ and for all $1 \leq m \leq n$, we have $a'_m \in Acts_m$, $(l_m, P_m(l_e), a'_m, l'_m) \in T_m$, and $(l_m, P_m(l_e)) \in y_j$ if $a'_m = a_j$ for some $1 \leq j \leq k$.
3. the relation $\sim'_m$ is defined as follows: $s \sim'_m s'$ iff either of the following conditions holds: (a) $s = s' = t'$; (b) $s = (l_e, ..., l_n, y_1, ..., y_k)$, $s' = (l'_e, ..., l'_n, y'_1, ..., y'_k)$, $l_m = l'_m$, $P_m(l_e) = P_m(l'_e)$, and $y_j = y'_j$ for all $1 \leq j \leq k$.
4. the function π' is defined as follows: $\pi'(t') = \emptyset$ and $\pi'((l_e, l_1, ..., l_n, y_1, ..., y_k)) = \pi(l_e, l_1, ..., l_n)$.
5. the fairness constraint $\mathcal{F}'$ is defined as follows: $(l_e, l_1, ..., l_n, y_1, ..., y_k) \in \mathcal{F}'$ iff $(l_e, l_1, ..., l_n) \in \mathcal{F}$.

Intuitively, all states are attached with a set of config-settings, the new transition relation is consistent with config-settings, and config-settings do not change along the transitions.

Let $M'(s)$ for $s \in S'$ be the same system with M' except for the initial state s, that is, $M'(s) = (S', s, T', \{\sim'_i\}_{i \in Agt}, \pi')$. First of all, the reconfigurability problem $M, A \models_{env} \phi$ is equivalent to $M'(s), \emptyset \models_{env} \phi$ for some state $s = (t_e, t_1, ..., t_n, y_1, ..., y_k)$ such that $\{y_1, ..., y_k\}$ is a set of config-settings of A.

Then for any state $s \in S'$, we consider the problem $M'(s), \emptyset \models_{env} \phi$. We proceed by induction on the formula ϕ. For $\phi = p \in V$, we have that $M'(s), \emptyset \models_{env} \phi$ iff $\phi \in \pi'(s)$.

For $\phi = x_{m,a}$ with $m \in Agt$ and $a \in LAct_m$, we have that $M'(s), \emptyset \models_{env} \phi$ with $s = (l_e, l_1, ..., l_n, y_1, ..., y_k)$ iff there exists $1 \leq j \leq k$ such that $agt(y_j) = m$, $act(y_j) = a$, and $(l_m, P_m(l_e)) \in y_j$, where $agt(y_j)$ and $act(y_j)$ represent the corresponding agent and action of the config-setting for y_j.

For $\phi = K_i\varphi$, we have $M'(s), \emptyset \models_{env} \phi$ iff $\forall \sigma : M'(s), \sigma, s \models \phi$. Then by the semantics of knowledge operator, the latter is equivalent to $\forall \sigma \forall s' \forall \sigma' : s \sim'_i s' \wedge s' \in rch(M'(s), \sigma') \Rightarrow M'(s), \sigma', s' \models \varphi$, which in turn is equivalent to $\forall s' : s \sim'_i s' \Rightarrow (\forall \sigma' : s' \in rch(M'(s), \sigma') \Rightarrow M'(s), \sigma', s' \models \varphi)$. Because σ' is any strategy with perfect recall memory, we have that $\forall s' : s \sim'_i s' \Rightarrow M'(s'), \emptyset \models_{env} \varphi$.

For ϕ being a CTL formula, we can check it by an automata theoretic approach. We use $T'(s) = \{s' \mid (s, s') \in T'\}$ to denote the set of successor states of s in system M'. Assume that $s = (l_e, l_1, ..., l_n, y_1, ..., y_k)$. Let $s', s'' \in T'(s)$ be two successor states of s such that $s' = (l'_e, l'_1, ..., l'_n, y_1, ..., y_k)$ and $s'' = (l''_e, l''_1, ..., l''_n, y_1, ..., y_k)$, we say that s' and s'' are in the same partition of $T'(s)$ if $l'_e = l''_e$ and there exist local actions $a_1, ..., a_n$ such that for all $i \in Agt$, there is $\{(l_i, P_i(l_e), a_i, l'_i), (l_i, P_i(l_e), a_i, l''_i)\} \subseteq T_i$. Intuitively, two states in the same partition of $T'(s)$ if they can be nondeterministically reached from s in a step after applying config-settings and environment strategy. That is, these are internal nondeterminism that cannot be eliminated by the interactions with the outside world.

From $M'(s) = (S', s, T', \{\sim'_i\}_{i \in Agt}, \pi')$, we define a Büchi tree automaton $A_{M'(s)} = (\Sigma, D, Q, \delta, q_0, Q)$ such that

1. $\Sigma = \mathcal{P}(V) \cup \{\bot\}$, $Q = S' \times \{\top, \vdash, \bot\}$, $q_0 = (s, \top)$,
2. $D = \bigcup_{s \in S} \{1, ..., |T'(s)|\}$,
3. $\delta : Q \times \Sigma \times D \to 2^{Q^*}$ is defined as follows: for $s' \in S'$ and $k = |T'(s')|$ with $T'(s') = (s_1, ..., s_k)$, we have (a) if $m \in \{\vdash, \bot\}$ then $\delta((s', m), \bot, k) = \{((s_1, \bot), ..., (s_k, \bot))\}$, and (b) if $m \in \{\vdash, \top\}$, then we let $((s_1, y_1), ..., (s_k, y_k)) \in \delta((s', m), \pi'(s'), k)$ such that, there exists a nonempty set $B \subseteq \{1, ..., k\}$ of indices such that for all $i, j \in B$, s_i and s_j are in the same partition of $T'(s')$, and

(a) $y_i = \top$, for all $i \in B$, and

(b) $y_j = \ \vdash$, for all $j \notin B$ and $1 \leq j \leq k$.

Note that we use $\mathcal{F} = Q$ to express that we only care about infinite paths. Moreover, the formula ϕ needs to be modified to reject those runs where $\bot$ is labeled on the states. This can be done by following the approach in [Kupferman and Vardi, 1996]. We still call the resulting formula ϕ.

Given a CTL formula ϕ and a set $D \subset \mathbb{N}$ with a maximal element k, there exists a Büchi tree automaton $A_{D,\neg\phi}$ that accepts exactly all the tree models of $\neg\phi$ with branching degrees in D. By [Vardi and Wolper, 1986], the size of $A_{D,\neg\phi}$ is $O(2^{k\cdot|\phi|})$. To decide whether $M'(s), \emptyset \models_{env} \phi$ for a CTL formula is equivalent to checking the emptiness of the product automaton $A_{M'(s)} \times A_{\neg\phi}$. The checking of emptiness of Büchi tree automaton can be done in quadratic time, so the checking of $M'(s), \emptyset \models_{env} \phi$ can be done in exponential time.

Therefore, checking whether $M'(s), \emptyset \models_{env} \phi$ can be done in exponential time with respect to $|S|$, $|\bigcup_{i\in Agt} Acts_i|$, and $|\phi|$. That is, the reconfigurability $M, A \models_{env} \phi$ is in EXPTIME.

We omit the proof for the lower bound, which is reduced from the problem of a linearly bounded alternating Turing machine accepting an empty input tape. □

Example 4 *(Continue with Example 3) The behaviour of the distributor is modelled by the environment. Therefore, when checking whether*

$$M, A \models_{env} \phi$$

for $\phi \equiv \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4 \wedge \phi_5 \wedge \phi_6 \wedge \phi_7$, it is actually checking reconfigurability with respect to the objective ϕ under all possible interactions of the distributor with the other modules of the bigger system. The interaction occurs by the inputs over the nondeterminism of the distributor on choosing agents to satisfy their requests.

The above example shows an additional advantage of reconfiguration with environment strategy over model checking in that, it can work directly with under-specified multiagent system and automatically verify the formulas without specifying the details of the aspect A2 (given in Section 3) for environment's protocol. As discussed in Section 3, both this advantage and the advantage from config-settings are essential for the reconfiguration of MCSs.

6 Related Work

Methodological research has been conducted on the specification of reactive multiagent systems for various applications, e.g., textile industry, vehicle collision avoidance [Yang *et al.*, 2008], transportation network [Meignan *et al.*, 2007], etc. The developments of these systems rely on the code developers, and the analysis of the systems is usually done by simulation techniques. Our approach is based on, and generalises, the logic-based specification and verification of multiagent systems, and therefore is rigorous and automated.

The interests towards the reconfiguration of systems are recent with prototype systems appearing, see e.g., [Bishop *et al.*, 2005; Oung *et al.*, 2010]. The reconfiguration actions in these system are hard-wired in their implementations, without rigorous proofs on their reliability, e.g, whether executing such an action at a specific system state is safe, or whether it is possible to executing such an action on a module of the system and at the same time ensuring the stability and predictability of the entire system. The paper complements this with the formalisation of reconfigurability, which is treated as a variant of model checking multiagent systems.

The techniques employed can be related to the module checking [Kupferman and Vardi, 1996], which considers the interactions of a monolithic system with outside world. There are major differences when considering multiagent systems. In module checking, the outside world is able to interact with all nondeterministic choices of the system. This contrasts with a multiagent system, in which an environment strategy can only interact with the environment protocol and cannot directly control agents' behaviour. To enable the interaction with agents' protocols and compete with environment strategies, config-settings are considered. Moreover, to work with multiple agents and their partial observations, we consider a logic CTLK($V, Agt, \bigcup_{i\in Agt} Acts_i$) in which we have knowledge operator K_i and can write structure $x_{i,a}$ to directly refer to the config-settings (See formulas ϕ_5, ϕ_6, ϕ_7 in Example 3 for such examples). These differences distance the concept of reconfigurablity with that of module checking. Recently, [Jamroga and Murano, 2014] discusses the difference between module checking and ATL model checking, and [Jamroga and Murano, 2015] discusses module checking based on ATL when environment is partial observation, which is different with our maximal assumption about the environment.

Reconfiguration can be regarded as a competitive game between agents and the environment. To reasoning about competitive games, various strategy logics, such as [Alur *et al.*, 2002; Huang and van der Meyden, 2014c; 2014a], have been proposed, together with model checking algorithms [Huang and van der Meyden, 2014b; Huang, 2015]. The paper works with a particular setting: a set of reactive agents operate their config-settings against an environment which have full information about the system and agents' strategy, to achieve a goal expressed with CTLK($V, Agt, \bigcup_{i\in Agt} Acts_i$) formula.

7 Conclusions and Future Work

In the paper, aiming at the applications in smart factories and Industrie 4.0, we formalise the reconfigurability for reactive multiagent systems, by generalising the approaches of model checking multiagent systems. Two reconfiguration scenarios are considered and their complexity results are given.

States and transitions of multiagent systems are high-level abstractions of the real-world manufacturing control systems. Working directly with concrete systems in which the states include geometric place information and the transitions are described by e.g., constraints in a three-dimensional space, is interesting. [Reif, 1979] considers the planning of a sequence of movements of linked polyhedra, which are suitable to precisely represent actual mechanical devices e.g., robot arms.

Moreover, we are also interested in stochastic multiagent systems, on which our efforts have been made towards probabilistic extensions of strategy logics [Huang *et al.*, 2012; Huang and Kwiatkowska, 2016; Huang and Luo, 2013]. Future work includes probabilistic reconfiguration with stochastic config-settings and strategies.

Acknowledgement The authors gratefully thank the reviewers for detailed and insightful comments. The authors are supported by ERC Advanced Grant VERIWARE, EPSRC Mobile Autonomy Programme Grant EP/M019918/1, National Natural Science Foundation of China Grant 61572234 and No.61472369, Fundamental Research Funds for the Central Universities of China Grant 21615441, and ARC Grant DP150101618.

References

[Alur *et al.*, 2002] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[Bishop *et al.*, 2005] J. Bishop, S. Burden, E. Klavins, and R. Kreisberg. Programmable parts: a demonstration of the grammatical approach to self-organization. In *IROS 2005*, pages 3684–3691, 2005.

[Chen *et al.*, 2006] I-Ming Chen, Guilin Yang, and Song Huat Yeo. Automatic modeling for modular reconfigurable robotic systems – theory and practice. In *Industrial Robotics: Theory, Modelling and Control*. Pro Literatur Verlag, Germany, 2006.

[Clarke *et al.*, 1999] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.

[Dennis *et al.*, 2014] Louise A. Dennis, Michael Fisher, Jonathan M. Aitken, Sandor M. Veres, Yang Gao, Affan Shaukat, and Guy Burroughes. Reconfigurable autonomy. *Künstliche Intelligenz*, 28(3):199–207, 2014.

[Fagin *et al.*, 1995] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[Gammie and van der Meyden, 2004] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *CAV2004*, pages 479–483, 2004.

[Huang and Kwiatkowska, 2016] Xiaowei Huang and Marta Kwiatkowska. Model Checking Probabilistic Knowledge: A PSPACE Case. In *AAAI 2016*, 2016.

[Huang and Luo, 2013] Xiaowei Huang and Cheng Luo. A Logic of Probabilistic Knowledge and Strategy. In *AAMAS 2013*, pages 845–852, 2013.

[Huang and van der Meyden, 2014a] Xiaowei Huang and Ron van der Meyden. An Epistemic Strategy Logic. In *SR 2014*, pages 35–41, 2014.

[Huang and van der Meyden, 2014b] Xiaowei Huang and Ron van der Meyden. Symbolic model checking epistemic strategy logic. In *AAAI 2014*, pages 1426–1432, 2014.

[Huang and van der Meyden, 2014c] Xiaowei Huang and Ron van der Meyden. A temporal logic of strategic knowledge. In *KR 2014*, 2014.

[Huang *et al.*, 2012] Xiaowei Huang, Kaile Su, and Chenyi Zhang. Probabilistic Alternating-time Temporal Logic of Incomplete information and Synchronous Perfect Recall. In *AAAI-12*, pages 765–771, 2012.

[Huang, 2015] Xiaowei Huang. Bounded model checking of strategy ability with perfect recall. *Artificial Intelligence*, 222:182–200, 2015.

[Ind, 2011] Industrie 4.0: Mit dem internet der dinge auf dem weg zur 4. industriellen revolution, 2011.

[Jamroga and Murano, 2014] Wojciech Jamroga and Aniello Murano. On module checking and strategies. In *AAMAS 2014*, pages 701–708, 2014.

[Jamroga and Murano, 2015] Wojciech Jamroga and Aniello Murano. Module checking of strategic ability. In *AAMAS 2015*, pages 227–235, 2015.

[Kasper Støy *et al.*, 2010] Kasper Støy, David Brandt, and David J. Christensen. *Self-Reconfigurable Robots*. MIT Press, 2010.

[Koren *et al.*, 2003] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel. Reconfigurable manufacturing systems. In *Manufacturing Technologies for Machines of the Future*, pages 627–665. Springer, 2003.

[Kupferman and Vardi, 1996] Orna Kupferman and Moshe Y. Vardi. Module checking. In *CAV 1996*, pages 75–86, 1996.

[Lomuscio *et al.*, 2009] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *CAV2009*, pages 682–688, 2009.

[Marik and McFarlane, 2005] V. Marik and D. McFarlane. Industrial adoption of agent-based technologies. *IEEE Intelligent Systems*, 20(1):IEEE Intelligent Systems, 2005.

[Meignan *et al.*, 2007] David Meignan, Olivier Simonin, and Abderrafiaa Koukam. Simulation and evaluation of urban bus-networks using a multiagent approach. *Simulation Modelling Practice and Theory*, 15(6):659–671, 2007.

[Nikolaus, 2014] Katrin Nikolaus. Manufacturing: Self-organizing factories, 2014.

[Oung *et al.*, 2010] R. Oung, F. Bourgault, M. Donovan, and R. D'Andrea. The distributed flight array. In *ICRA2010*, pages 601–607, 2010.

[Paulo Leitão, 2009] Paulo Leitão. Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7):979–991, 2009.

[Reif, 1979] John H. Reif. Complexity of the mover's problem and generalizations. In *FOCS 1979*, pages 421–427, 1979.

[Tang and Wong, 2005] H.P. Tang and T.N. Wong. Reactive multi-agent system for assembly cell control. *Robotics and Computer-Integrated Manufacturing*, 21(2):87–98, 2005.

[Vardi and Wolper, 1986] Moshe Y. Vardi and Pierre Wolper. Automata-theoretic techniques for modal logics of programs. *J. Comput. Syst. Sci.*, 32(2):183–221, 1986.

[Yang *et al.*, 2008] Sibo Yang, F. Gechter, and A. Koukam. Application of reactive multi-agent system to vehicle collision avoidance. In *ICTAI 2008*, pages 197 – 204, 2008.

[Zaske, 2015] Sara Zaske. Germany's vision for industrie 4.0: The revolution will be digitised, 2015.