Simple Knowledge-based Programs in *Partially-Known* Environment

Abstract

Knowledge-based programs (KBPs) is a high-level multiagent program specification. In this paper, inspired by the modelling of systems such as robot swarms and robotic search-and-rescue, we relax two implicit assumptions in KBPs' original semantics. Specifically, we consider multiagent systems in which (1) agents are simple instead of rational, and/or (2) the environment is partially-known to the agents instead of completely-known. For the latter, we further consider two cases in which agents either passively maintain or actively maintain its knowledge about the environment. We present new semantics for the relaxed assumptions and study complexity of the KBP realisability problem, with the following main results: (1) the assumption of simple agents eliminates the exponential blowup of the complexity with respect to the number of agents, and (2) the assumption of partially-known environment does not change the complexity, irrespectively of agents passively or actively maintain their knowledge. Therefore, our relaxation of assumptions comes for free or leads to lower complexity. We also discuss how to re-use existing tool to implement the new semantics.

Introduction

Knowledge-based programs (KBPs) [Fagin *et al.*, 1997] is a well-founded high-level program specification. Many of the interesting analyses of problems in multiagent computing based on notions of knowledge (e.g. [Halpern and Zuck, 1992]) can be cast in the form of knowledge-based programs. In a multiagent system, agents are autonomous, meaning that they have the ability to operate and adapt to changing circumstances with reduced or without human control [NSTC, 2016]. When specifying a multiagent system as a KBP, each agent has an abstract protocol describing the interaction of its locally available data, formalised as formulas expressing agent's knowledge, and its local actions.

The paper is inspired by the modelling of multiagent systems, such as robot swarms and robotic search-and-rescue, with KBPs. Robot swarms study the coordination of multirobot systems which consist of large numbers of mostly simple physical robots[1]. There are three commonly-made assumptions for swarm robots. The first is that, agents are *simple*, which means that, though they are able to observe the environment with sensors and communicate with other agents with communication devices such as WiFi, they have simple behaviour and do not consider other agents' strategies when making decisions. This contrasts with rational agents studied in various solution concepts in computational game theory where agents need to consider each other's strategies to reach equilibriums. The second is that, the environment is *partially-known* to an agent, instead of completely known. More specifically, agents may not be aware of the existence of some information (e.g., other agents, part of the map, etc). This needs to be differentiated with partial observation, with which agents know the existence of an information but do not know its value. The partial observation assumption is also made throughout this paper. The third is that, agents are *reactive*, i.e., they may not be able to remember their observation histories in their memory. Instead, they have to make decisions based on limited information, or more specifically, the information they observe in the current round. We emphasis that, in real world, there are many other systems which present these three characteristics. Typical examples include a set of self-driving cars on a road, a set of robots in a smart factory [GTAI, 2014], and some robotic search-and-rescue scenarios, etc.

A further classification based on partially-known environment assumption is that agents may be *passively maintaining* its knowledge about the environment, i.e., they only maintain the information provided by the environment in the current time, or *actively maintaining* its knowledge, i.e., they maintain the information it has been provided up to the current time. The maintaince is done by updating a finite structure representing part of the environmental information (such as the map or the number of agents). This needs to be differentiated with perfect recall [Fagin *et al.*, 1995], in which a (possibly infinite) sequence of observations are maintained. Notable examples for the passive maintanence include robot swarms, and for the active maintanence robotic search-and-rescue in [Ayala *et al.*, 2013].

As a high-level specification, KBPs are not directly executable for their use of epistemic logic formulas (in this paper

[1]In the following, we call each robot an agent.

CTLK [Fagin *et al.*, 1995]) in agents' protocols. Concrete implementations of the protocols are needed by replacing epistemic logic formulas with semantically equivalent propositional logic formulas. In its original semantics, such *KBP realisability problem* implicitly works with the assumptions of rational agents and completely-known environment, which, as explained, are not suitable for some systems.

This paper has two main contributions. The first main contribution is to present new semantics for KBPs under the assumptions of simple agents and/or partially-known environment. The new semantics is applied to explain a specific system of swarm robots from [Kouvaros and Lomuscio, 2015; Nembrini, 2005] and we also discuss the case of robotic search-and-rescue from [Ayala *et al.*, 2013]. The realisability problem of the new semantics can be implemented by reusing the existing tool for the original semantics.

The second main contribution is to study the changes of computational complexity of the KBP realisability problem under the new semantics. For original semantics, it is NP-complete with respect to the size $|M|$ of the system and PSPACE-complete with respect to the number $|Agt|$ of agents. When considering simple agents, the complexity is lowered to PTIME-complete with respect to $|Agt|$. When considering partially-known environment with either active or passive maintanence, we find that the complexity does not change. These results are somewhat unexpected because usually the relaxation of assumptions results in higher computational complexity. The results for partially-known environment with active maintanence are the most surprising: the complexity remains in NP (instead of moving up to PSPACE) even if every agent constantly maintains a structure of polynomial size to represent its awareness about the environment.

Preliminaries

Let $\mathcal{B}(Var)$ be the set of boolean formulas over variables Var. Without loss of generality, we assume that all variables are boolean. For s being a truth assignment of the variables Var and $f \in \mathcal{B}(Var)$ a formula, we write $e(s, f)$ for the evaluation of f on s. We may write $e(s, f)$ (or $\neg e(s, f)$) to denote that $e(s, f) = 1$ ($e(s, f) = 0$).

A multi-agent system consists of a collection of agents running in an environment [Fagin *et al.*, 1995]. The environment E is a tuple $(Agt, Var_e, init_e, \{Acts_i\}_{i\in Agt}, \{OVar_i\}_{i\in Agt}, T_e)$, where $Agt = \{1, ..., n\}$ is a set of agents. The component Var_e is a set of environment variables such that every truth assignment to Var_e is an environment state. Let L_e be the set of environment states. The component $init_e \subseteq L_e$ is a set of initial environment states, $OVar_i \subseteq Var_e$ is a subset of environment variables that agent i is able to observe, $Acts_i$ is a set of local actions for agent i such that $Acts_i \cap Acts_j = \emptyset$ if $i \neq j$ and $JActs = \times_{i\in Agt} Acts_i$ is a set of joint actions, and $T_e \subseteq L_e \times JActs \times L_e$ is a transition relation. The environment *nondeterministically* updates its state by taking into consideration the joint actions taken by the agents. Agents' observable variables may be overlapping, i.e., $OVar_i \cap OVar_j \neq \emptyset$, to simulate the case where agents have shared variables. We use $s_{e,i}$ to denote the part of an environment state s_e that can be observed by agent i, i.e., $s_{e,i} = s_e|_{OVar_i}$. This can be generalized to a set of states, e.g., $L_{e,i} = \{s_{e,i} \mid s_e \in L_e\}$.

An agent A_i, for $i \in Agt$, is a tuple $(Var_i, init_i, T_i)$. The component Var_i is a set of local variables such that each truth assignment to Var_i is a local state. Let L_i be the set of local states of agent i. The component $init_i \subseteq L_i$ is a set of initial local states, $T_i \subseteq L_i \times L_{e,i} \times Acts_i \times L_i$ is a transition relation: a tuple $(l_i, o_i, a_i, l'_i) \in T_i$ means that when agent i is at state l_i and has an observation o_i on the environment state, it may take action a_i and move into the state l'_i. If there are several a_i with the same l_i and o_i, the agent i will *nondeterministically* choose one of them to execute.

Let $Var = Var_e \cup \bigcup_{i\in Agt} Var_i$ and $Acts = \bigcup_{i\in Agt} Acts_i$. A multi-agent system is defined as $M(E, \{A_i\}_{i\in Agt}) = (S, I, T, \{O_i\}_{i\in Agt})$. The set $S = L_e \times \Pi_{i\in Agt} L_i$ is a set of global states. For a global state $s = (l_e, l_1, ..., l_n)$, we write $s_i \equiv l_i$ for $i \in Agt$, and $s_e \equiv l_e$. The same for joint actions. The set I is a set of initial states such that $s \in I$ if $s_e \in init_e$ and $s_i \in init_i$ for all $i \in Agt$. The transition relation $T \subseteq S \times JActs \times S$ is defined as $(s, a, t) \in T$ if $(s_i, s_{e,i}, a_i, t_i) \in T_i$ for all $i \in Agt$ and $(s_e, a, t_e) \in T_e$. The observation functioin $O_i : S \to L_i \times L_{e,i}$ is such that $O_i(s) = (s_i, s_{e,i})$. We use $N_i((s_i, s_{e,i})) = \{a_i \in Acts_i \mid \exists t_i \in L_i : (s_i, s_{e,i}, a_i, t_i) \in T_i\}$ to denote the set of local actions of agent i that are enabled on global state s. We assume that the environment transition relation T_e is serial, i.e., for every state s and every joint action a such that $a_i \in N_i((s_i, s_{e,i}))$ for all $i \in Agt$, there exists a state t such that $(s, a, t) \in T_e$. However, we do not assume the same for agents, i.e., given a local state s_i and an observation $s_{e,i}$, a local action a_i may be disabled.

A (labelled) path $\rho = s^0 a^1 s^1 ...$ is a finite or an infinite sequence of states such that s^0 is an initial state, and for every $k \geq 0$, $(s^k, a^{k+1}, s^{k+1}) \in T$. We use $\rho(m)$ to denote the state s^m. Moreover, we write $Path^F_M(s)$ ($Path^I_M(s)$, resp.) for the set of finite (infinite) paths ρ of M such that $\rho(0) = s$. For a finite path $\rho \in Path^F_M(s)$, its length $|\rho|$ is the number of states on ρ and we write $last(\rho)$ for its last state. Let $\rho[0..k] = s^0 ... a^k s^k$ be the prefix of ρ up to time k. Moreover, we write $Path^I_M$ and $Path^F_M$ for the set of paths $\bigcup_{s\in I} Path^I_M(s)$ and $\bigcup_{s\in I} Path^F_M(s)$, respectively.

Strategies A (memoryless) strategy of agent i in M can be represented as a function $\theta_{M,i} : L_i \times L_{e,i} \to \mathcal{P}(Acts_i) \setminus \{\emptyset\}$ mapping from current observations to nonempty sets of local actions such that $\theta_{M,i}((s_i, s_{e,i})) \subseteq N_i((s_i, s_{e,i}))$ for all $s \in S$. Let Γ_i be the set of strategies of agent i. A path $\rho = s^0 a^1 s^1 ...$ is consistent with a strategy $\theta_{M,i} \in \Gamma_i$ if for all $k \geq 0$ we have $a^{k+1}_i \in \theta_{M,i}((s^k_i, s^k_{e,i}))$. Let $\Gamma = \times_{i\in Agt} \Gamma_i$ be the set of joint strategies (or strategy profiles). Given a strategy profile $\theta_M = \{\theta_{M,i}\}_{i\in Agt}$, the sets $Path^F_M(s), Path^I_M(s), Path^F_M$ and $Path^I_M$ can be restricted to $Path^F_M(\theta_M, s), Path^I_M(\theta_M, s), Path^F_M(\theta_M)$ and $Path^I_M(\theta_M)$ to include only paths consistent with $\theta_{M,i}$ for all $i \in Agt$.

Given two paths ρ_1 and ρ_2 such that $last(\rho_1) = \rho_2(0)$, we write $\rho_1 \cdot \rho_2$ for the concatenation of them by the overlapping state $last(\rho_1)$. Moreover, given a model M and a strategy profile θ_M, we write $rch_M(\theta_M)$ for the set of reachable states of M under θ_M, i.e., $s \in rch_M(\theta_M)$ if there exists a state $t \in I$ such that there exists an infinite path $\rho \in Path^I_M(\theta_M, t)$ such that $s = \rho(m)$ for some $m \geq 0$.

Specification Language We use a language CTLK to describe the specifications of a multi-agent system M. Formally, CTLK has the syntax:

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

where $v \in Var$ and $i \in Agt$. Intuitively, formula $K_i\phi$ means that agent i knows ϕ. Temporal formulas follow the standard meanings as in [Clarke *et al.*, 1999]. Other operators can be obtained in the usual way, e.g., $A\phi = \neg E \neg \phi$, $F\phi = True \mathrm{U} \phi$.

The semantics of the language on a system M and a strategy profile $\theta_M \in \Gamma$ is described by recursively defining a relation $M, \theta_M, s \models \phi$, for $s \in S$ a global state, over the structure of the formula ϕ as follows.

- $M, \theta_M, s \models v$ if $e(s, v)$.
- boolean operators follow the standard definition.
- $M, \theta_M, s \models K_i\phi$ if for all states $s' \in rch_M(\theta_M)$ such that $O_i(s) = O_i(s')$, we have $M, \theta_M, s' \models \phi$.
- $M, \theta_M, s \models EX\phi$ if $M, \theta_M, s' \models \phi$ for some state s' and joint action a such that $(s, a, s') \in T$ and $a_i \in \theta_{M,i}((s_i, s_{e,i}))$ for all $i \in Agt$.
- $M, \theta_M, s \models E(\phi_1 \mathrm{U} \phi_2)$ if there exist $\rho \in Path^I_M(\theta_M, s)$ and $m \geq 0$ such that $M, \theta_M, \rho(m) \models \phi_2$ and $M, \theta_M, \rho(k) \models \phi_1$ for all $0 \leq k \leq m-1$.
- $M, \theta_M, s \models EG\phi$ if there exists $\rho \in Path^I_M(\theta_M, s)$ such that $M, \theta_M, \rho(k) \models \phi$ for all $k \geq 0$.

Given a multi-agent system M, a joint strategy θ_M, and a formula ϕ, the model checking problem, written as $M, \theta_M \models \phi$, is to decide whether $M, \theta_M, s \models \phi$ for all $s \in I$[2].

Knowledge-based Programs

Given a multiagent system M, a knowledge-based program (KBP) [Fagin *et al.*, 1997] can be regarded as additional specification of M describing the interaction of agents' knowledge with their actions. Formally, each agent i has a protocol Q_i of the form

$$\texttt{do } K_i\phi_{i,1} \rightarrow a_{i,1} \; [] \; \ldots [] \; K_i\phi_{i,y_i} \rightarrow a_{i,y_i} \texttt{ od}$$

where for all $1 \leq j \leq y_i$ we have $a_{i,j} \in Act_i$, and each $a_i \in Act_i$ appears just once. Intuitively, Q_i says to repeat forever the following operation: nondeterministically execute one of the actions $a_{i,j}$ whose guard $K_i\phi_{i,j}$ is true. Based on M, a KBP is a collection of protocols $Q = \{Q_i\}_{i \in Agt}$, one for each agent.

A KBP is an abstract program that cannot be directly executed by agents because of the knowledge guards, whose values cannot be decided locally. We define the original semantics of KBPs through *strategic reasoning*. The definition is different with that of [Fagin *et al.*, 1997] but is nonetheless equivalent to theirs according to [Huang and van der Meyden, 2014a]. To make KBP executable, a concrete implementation is needed. A concrete implementation can be represented as a strategy profile $\theta_M = \{\theta_{M,i}\}_{i \in Agt}$ such that

$$M, \theta_M, s \models AG \bigwedge_{i \in Agt} \bigwedge_{j=1}^{y_i} (K_i\phi_{i,j} \Leftrightarrow O_{i,j}) \text{ for } s \in S \qquad (1)$$

[2]We have a slightly different definition of model checking problem by having θ_M. This is to be useful for explaining KBPs.

where $O_{i,j} = \{(s_i, s_{e,i}) \mid a_{i,j} \in \theta_{M,i}((s_i, s_{e,i}))\}$ is a set of observations on which the action $a_{i,j}$ is enabled under strategy $\theta_{M,i}$. It is noted that the set $O_{i,j}$ can be represented as a formula through the variables $Var_i \cup OVar_i$. If such a strategy profile θ_M exists, the concrete and executable implementation Q'_i of Q_i is as follows by replacing knowledge formulas $K_i\phi_{i,j}$ with the propositional logic formulas representing $O_{i,j}$:

$$\texttt{do } O_{i,1} \rightarrow a_{i,1} \; [] \; \ldots [] \; O_{i,y_i} \rightarrow a_{i,y_i} \texttt{ od}$$

The KBP realisability problem is, given a multiagent system $M(E, \{A_i\}_{i \in Agt})$ and a KBP $Q = \{Q_i\}_{i \in Agt}$, to determine whether there exists a strategy profile θ_M such that the expression (1) is satisfiable. The following theorem gives the computational complexity of KBP realisability problem (with respect to the specification language CTLK). We assume that the size $|M|$ of the system is measured with the number $|S|$ of states.

Theorem 1 *Given a multiagent system $M(E, \{A_i\}_{i \in Agt})$ and a KBP $Q = \{Q_i\}_{i \in Agt}$, the KBP realisability problem for strategies in Γ is NP-complete with respect to $|M|$ and PSPACE-complete with respect to the number $|Agt|$ of agents.*

The NP-completeness can be obtained by the guess-and-verify approach as that of [Vardi, 1996] (for upper bound) and a reduction from 3SAT problem (for lower bound). For PSPACE-completeness, the upper bound can be obtained by a nondeterministic algorithm taking a polynomial size of space to store a strategy for each agent and the lower bound can be obtained by a reduction from the emptiness problem of linear bounded automata which is known to be PSPACE-complete[3].

Illustrative Example

A detailed investigation into Equation (1) suggests that two implicit assumptions are made on the original KBP semantics. The first is that, the strategy profile θ_M is a common knowledge of the agents. Such assumption is reasonable for rational agents, i.e., agents need to make rational decisions and each of their satisfiable strategy profiles is an equilibrium point in the joint strategy space. In [Halpern and Moses, 2007], it has been shown that KBPs can be adapted to describe solution concepts. However, for some systems, agents do not follow rational decision, but decide their behaviour with respect to the information they received, i.e., an agent does not consider other agents' strategies when making decisions. Such systems include robot swarms, which is to be modelled below, self-driving scenarios, in which all cars in autonomous driving mode decide their behaviour by reading signals from its sensors, and some robotic search-and-rescue, in which agents coordinate by communication.

The second implicit assumption is that, the system M is completely known (and is also a common knowledge) to all agents. This is an usual assumption for model checking multiagent systems, but can be nontheless unreasonable for many practical systems. In robot swarms, an agent may not know the existence of some agents; in self-driving scenarios, a car

[3]All the proofs in the paper are in the longer version of the paper, omitted in this submission due to space limit.

can only know the exisence of those cars within the capability of its sensors and moreover may not know the entire map and traffic information; in search-and-recue, an agent may not know e.g., the entire map and the existence of obstacles.

In this paper, we will relax these two assumptions. In this section, we will introduce an example of robot swarms from [Kouvaros and Lomuscio, 2015], which is taken from [Nembrini, 2005]. The cases of simple agents and partially-known enviroment with passive maintainence will be studied respectively in the next two sections on this example, followed by a section explaining the partially-known enviroment with active maintainence. The concepts of passive and active maintainence will be introduced in relevant sections.

Example 1 *Given a two-dimensional grid G of size* $(\alpha+1)\times(\alpha+1)$ *for* $\alpha \geq 0$. *We use pairs* (x, y) *to range over the grid points such that* $0 \leq x, y \leq \alpha$. *There are a set of n robots, each of which stays in one of the grid points* $(x_i, y_i) \in [0..\alpha]\times[0..\alpha]$ *and has a direction* $d_i \in \{e(ast), w(est), n(orth), s(outh)\}$ *for a potential movement.*

Each robot has a communication range $\beta \geq 0$ *to communicate with its neighbours within the range. Two robots i and j are neighbours, written as* $i \frown j$, *if*

$$\min\{|x_i - x_j|, \alpha - |x_i - x_j|\} \leq \beta \text{ and } \min\{|y_i - y_j|, \alpha - |y_i - y_j|\} \leq \beta \quad (2)$$

The behaviour of each robot can be described as follows. A robot i loses (the communication with) another robot j if $i \frown j$ *holds in the previous round but not in the current round. If the number of agent i's neighbours that can observe a lost robot in their neighbourhoods is less than* β *then i will make a* $180°$ *turn. Otherwise, if the number of agent i's neighbours increase then the robot makes a random* $90°$ *turn. If both the above conditions do not satisfy, i moves forward one cell along its current direction.*

For the environment E, we have the following variables

$$Var_e = \{nb_{i,j}, oldnb_{i,j}, nbl_{i,j} \mid i, j \in Agt\} \cup \{p_i, d_i, l_i \mid i \in Agt\}$$

where $nb_{i,j}, oldnb_{i,j} \in \{0, 1\}$ *represent whether agents i and j are neighbours in the current round and in the previous round, respectively,* $l_i \in \{0, 1\}$ *represents whether agent i has a lost neighbour,* $nbl_{i,j}$ *represents whether agents i and j are neighbours and agent j has a lost neighbour, and* p_i *and* d_i *represent agent i's position and direction, respectively. We let* $OVar_i = \{nb_{i,j}, oldnb_{i,j}, nbl_{i,j} \mid j \in Agt\}$ *be the set of environment variables that are observable to agent i. Note that, the agent does not have to observe its absolute position* p_i *and diretion* d_i. *The set* $init_e$ *includes a single initial state in which variables* l_i *and* $nbl_{i,j}$ *are* 0 *and other variables are set to appropriate values for the initial scenario.*

Each agent $i \in Agt$ *has a set* $Acts_i = \{move, t180, l90, r90\}$, *where* $t180$ *means that agent will turn* $180°$, $l90$ *and* $r90$ *mean that agent will turn* $90°$ *in the left and right direction, respectively, and move means that agent will move forward for one cell. The transition relation* T_e *can be described with the following sequential steps. First of all, given a joint action, agents' positions and directions will be updated and the values for variables* $nb_{i,j}$ *are stored into variables* $oldnb_{i,j}$. *Then according to the new positions, we can update variables* $nb_{i,j}$ *with respect to Equation (2). The variables* l_i *and* $nbl_{i,j}$ *are updated with the following expression*

$$l_i = \vee_{j\in Agt}(oldnb_{i,j} \wedge \neg nb_{i,j}) \text{ and } nbl_{i,j} = nb_{i,j} \wedge l_j \quad (3)$$

For every agent $i \in Agt$, *we have a simple protocol* A_i *that* $Var_i = \emptyset$ *and* $T_i = \{(\emptyset, a, \emptyset)\}$ *for* $a \in Acts_i$.

Before introducing knowledge-based protocols for agents, we define several notations as follows:

- $nbl_i(\geq k) \equiv \bigvee_{Q\subseteq Agt,|Q|=k} \bigwedge_{j\in Q}(nb_{i,j} \wedge l_j)$
- $nb_i(\geq k) \equiv \bigvee_{Q\subseteq Agt,|Q|=k} \bigwedge_{j\in Q} nb_{i,j}$
- $oldnb_i(< k) \equiv \neg \bigvee_{Q\subseteq Agt,|Q|\geq k} \bigwedge_{j\in Q} oldnb_{i,j}$

Intuitively, $nbl_i(\geq k)$ *expresses that agent i has at least k neighbours who lost at least one neighbours,* $nb_i(\geq k)$ *expresses that agent i has at least k neighbours, and* $oldnb_i(< k)$ *expresses that agent i has less than k neighbours in the previous round. Then the protocol* Q_i *for agent i is as follows:*

$$\begin{array}{lll} \texttt{do} \; K_i(\neg nbl_i(\geq \beta)) & \rightarrow & t180° \\ \quad \neg K_i(\neg nbl_i(\geq \beta)) \wedge K_i increase_i & \rightarrow & l90° \\ \quad \neg K_i(\neg nbl_i(\geq \beta)) \wedge K_i increase_i & \rightarrow & r90° \\ \quad otherwise & \rightarrow & move \;\; \texttt{od} \end{array}$$

where $increase_i \equiv \bigvee_{k=1}^{|Agt|}(nb_i(\geq k) \wedge oldnb_i(< k))$ *and the guard otherwise has the obvious meaning.*

Simple Agents

The first dimension we consider from knowledge-based programs is to let agents work loosely without considering each others' knowledge-based protocols. Formally, we get the implementation (i.e., a strategy $\theta_{M,i}$) for each agent i with respect to the following equation

$$M, \theta^{\perp}_{M,i}, s \models AG \bigwedge_{j=1}^{y_i} (K_i\phi_{i,j} \Leftrightarrow O_{i,j}) \text{ for } s \in S \quad (4)$$

where $\theta^{\perp}_{M,i}$ is a strategy profile in which agent i follows $\theta_{M,i}$ and other agents follow trivial strategy $\theta_{\perp}$. A trivial strategy for agent j is such that $\theta_{\perp}((s_j, s_{e,j})) = N_j((s_j, s_{e,j}))$ for all states s. We emphasise that, though simple agents do not consider each other's strategies, they can interact through shared environment variables. Shared variables can be used to model communication.

The following theorem shows that two semantics are equivalent for the robot swarm example.

Theorem 2 *The semantics for simple agents as in (4) is the same as the original semantics in (1) for robot swarms in Example 1.*

However, this conclusion is not general. Let $M\theta_M$ be the system in which all behaviour has to be consistent with θ_M. It is noted that generally the system $M\theta^{\perp}_{M,i}$ has more behaviour than $M\theta_M$ because only agent i has a non-trivial strategy to follow. This is reasonable for robot swarms: when agent i considers its protocol implementation, it may not have knowledge about the other agent j's protocol implementation or even abstract protocol. For the system in Example 1, agent j can be outside of the communication range of i and agent i does not know the existence of j.

The following theorem presents the complexity of the KBP realisability problem after taking simple agents.

Theorem 3 *Given a multiagent system $M(E, \{A_i\}_{i\in Agt})$ and a KBP $Q = \{Q_i\}_{i\in Agt}$, when assuming simple agents, the KBP realisability problem for strategies in Γ is NP-complete with respect to $|M|$ and in PTIME w.r.t. the number $|Agt|$ of agents.*

Except for the conceptual suitability for systems such as those discussed before, simple agents have computational advantage: the exponential blowup with respect to the number of agents is eleminated.

Partially Known Enviro.: Passively Maintained

In a partially-known environment, agents do not know what exactly the environment E is. Instead, each agent i is associated with a structure representing its awareness of some part of the environment. The structure is represented by restricting E to the sub-environment E^i paramterised over a subset Agt^i of agents and a subset Var_e^i of environmental variables.

Formally, given $Agt^i \subseteq Agt$ and $Var_e^i \subseteq Var_e$, we define $E(Agt^i, Var_e^i) = (Agt^i, Var_e^i, init_e{}^i, \{Acts_j\}_{j\in Agt^i}, OVar_i, T_e^i)$. The states L_e^i is the set of truth assignments to Var_e^i, $JActs^i = \times_{j\in Agt^i} Acts_j$ is a set of joint actions, and $T_e^i \subseteq L_e^i \times JActs^i \times L_e^i$ is a transition relation. Moreover, we have $init_e{}^i = L_e^i$, i.e., all states are initial states. Let Ω^i be the set of such structures. Apparently, $E \in \Omega^i$. In the following, we use E^i to range over Ω^i and let $M^i = M(E^i, \{A_i\}_{i\in Agt^i})$.

Let $Agt(E^i)$ and $Var_e(E^i)$ be the set of agents and environment variables in the structure E^i. Given a state s and a structure E^i, we write $s|_{E^i}$ for the part of state that is defined over the environment variables $Var_e(E^i)$ and agents $Agt(E^i)$.

Unlike M, the structure E^i is not static and will be updated with the system evolves. We consider two update approaches suitable for different applications. The first approach, to be considered in this section, *passively maintains* E^i according to the information available in the current time. The second approach will be discussed in the next section.

Both approaches employ a generic update function $u : \Omega^i \times S \to \Omega^i$ which takes the existing structure E^i and a state s and updates into a new structure $u(E^i, s)$. We also write $u(M^i, s)$ for $M(u(E^i, s), \{A_i\}_{i\in Agt(u(E^i,s))})$. Before proceeding to the instantiation of function u, we lift the relation $M, \theta_M, s \models \phi$ into $M, M^i, \theta_{M^i}, s \models \phi$ and define the new semantics as follows.

- $M, M^i, \theta_{M^i}, s \models v$ if $e(s, v)$.
- boolean operators follow the standard definition.
- $M, M^i, \theta_{M^i}, s \models K_i\phi$ if for all states $s' \in rch_{M^i}(\theta_{M^i})$ such that $O_i(s) = O_i(s')$, we have $M, M^i, \theta_{M^i}, s' \models \phi$.
- $M, M^i, \theta_{M^i}, s \models EX\phi$ if $M, u(M^i, s'), \theta_{u(M^i,s')}, s' \models \phi$ for some state $s' \in L_e^i$ and joint action $a \in JActs^i$ such that $(s, a, s') \in T^i$ and $a_i \in \theta_{M^i,i}((s_i, s|_{E^i}))$ for all $i \in Agt$.
- $M, M^i, \theta_{M^i}, s \models E(\phi_1 \mathrm{U} \phi_2)$ if $M, M^i, \theta_{M^i}, s \models \phi_2 \vee (\phi_1 \wedge EXE(\phi_1 \mathrm{U} \phi_2))$.
- $M, M^i, \theta_{M^i}, s \models EG\phi$ if $M, M^i, \theta_{M^i}, s \models \phi \wedge EXEG\phi$

Intuitively, for the case of $EX\phi$, the structure M^i is updated into $u(M^i, s')$ when moving to a new state s', and $(s_i, s|_{E^i})$ is the local state of agent i in M^i. The strategy profile θ_{M^i} is also changed into $\theta_{u(M^i,s')}$. Moreover, unlike the definition for $M, \theta_M, s \models \phi$, we use the standard iterative definitions of EU and EG [Clarke *et al.*, 1999] to avoid the technicalities involved in updating the structure along the path.

Different with the previous semantics in which an implementation for an abstract protocol Q_i is a strategy, an implementation under the new semantics is a set of strategies $\{\theta_{M^i,i}\}_{E^i\in\Omega^i}$, one for each structure $E^i \in \Omega^i$. Let $\Gamma_{E^i,i}$ be the set of i's strategies on M^i. An implementation needs to satisfy the following equation

$$M, M^i, \theta^{\perp}_{M^i,i}, s \models AG \bigwedge_{j=1}^{y_i} (K_i\phi_{i,j} \Leftrightarrow O_{i,j}) \text{ for } s \in S^i, E^i \in \Omega^i \tag{5}$$

where $\theta^{\perp}_{M^i,i}$ represents the strategy profile in which agent i takes $\theta_{M^i,i}$ and other agents take the trivial strategy $\theta_{\perp}$.

Now we proceed to the specific semantics for passive maintainence. Simply speaking, passive maintainence means that E^i is dependent on the current state s. We use functions $ev_M^i : S \to \mathcal{P}(Var_e)$ and $eg_M^i : S \to \mathcal{P}(Agt)$ to represent the subset of agents and environment variables that are exposed to agent i on a particular state, respectively. Then for the update function, we let

$$u(E^i, s) = E(eg_M^i(s), ev_M^i(s)) \tag{6}$$

Example 2 *(Continue with Example 1) For every state s, we let $eg_M^i(s) = \{j \mid nb_{i,j} = 1 \vee oldnb_{i,j} = 1\}$, representing the set of agents which are i's neighbours of either this round or the previous round, and $ev_M^i(s) = \{nb_{i,j}, oldnb_{i,j}, nbl_{i,j} \mid j \in Agt^i\} \cup \{p_i, d_i, l_i\}$, representing those variables that agent i should be aware of. Intuitively, an agent i will assume the non-existence of the other agent j if the latter is out of its sight for two consecutive rounds.*

Now we have the complexity results when combining with the assumption of simple agents.

Theorem 4 *Given a multiagent system $M(E, \{A_i\}_{i\in Agt})$ and a KBP $Q = \{Q_i\}_{i\in Agt}$, when assuming simple agents and partially-known environment with passive maintainence, the KBP realisability problem for strategies in Γ is NP-complete with respect to $|M|$ and in PTIME with respect to $|Agt|$.*

If combining with rational agents, we have the following.

Theorem 5 *Given a multiagent system $M(E, \{A_i\}_{i\in Agt})$ and a KBP $Q = \{Q_i\}_{i\in Agt}$, when assuming rational agents and partially-known environment with passive maintainence, the KBP realisability problem for strategies in Γ is NP-complete with respect to $|M|$ and PSPACE-complete w.r.t. $|Agt|$.*

Comparing with Theorem 1 and 3 for completely-known environment, the assumption of partially-known environment with passive maintainence does not change the complexity.

Partially-Known Enviro.: Actively Maintained

In this section, we consider the other update approach on the structure E^i where agents *actively maintain* E^i. We define the following update function

$$u(E^i, s) = E(Agt(E^i) \cup eg_M^i(s), Var_e(E^i) \cup ev_M^i(s)) \tag{7}$$

which means that agent i keeps track of all those agents and environment variables it has been aware of up to now.

Such setting is suitable for some robotic search-and-rescue as described in e.g., [Ayala *et al.*, 2013]. Initially the robot knows nothing about the environment except its current cell and the set of labels it may encounter within the environment. Then, the robot gains information about the environment by means of its sensors. Every time a new sensor reading is obtained, all the cells within the sensors range are updated and integrated into the robots knowledge. Such an update includes the accurate identification of the labels corresponding to each cell within the sensors range, as well as their occupancy condition. The robot maintains the currently known map and the history of each one of the cells it has moved through starting from the initial cell. Technically, we have that the structure E^i is constantly maintained by gradually adding more variables from $ev^i_M(s)$ and more agents from $eg^i_M(s)$, for s being the current state. This is formalised in the update function in Equation (7).

Surprisingly, as given below, the complexities are the same as the cases of passive maintainence, even though a structure E^i of polynomial size has to be explicitely maintained along every system execution. Usually, such an explicit maintainence of structures of polynomial size leads to the complexity class PSPACE (see e.g., the LTLK model checking for a single agent with perfect recall [van der Meyden and Shilov, 1999], in which the agent maintains a set of states that it thinks are possbile). However, we notice that, the number of such structures is at most $|S|$, i.e., $O(|\Omega^i|) = O(|S|)$. This enables the empolyment of a non-deterministic algorithm to guess for a polynomial number of times (to get a sequence of finite structures) along every execution, and therefore keeps the complexity in NP.

If combining with simple agents, we have the following.

Theorem 6 *Given a multiagent system $M(E, \{A_i\}_{i\in Agt})$ and a KBP $Q = \{Q_i\}_{i\in Agt}$, when assuming simple agents and partially-known environment with active maintainence, the KBP realisability problem for strategies in Γ is NP-complete with respect to $|M|$ and in PTIME with respect to $|Agt|$.*

If combining with rational agents, we have the following.

Theorem 7 *Given a multiagent system $M(E, \{A_i\}_{i\in Agt})$ and a KBP $Q = \{Q_i\}_{i\in Agt}$, when assuming rational agents and partially-known environment with active maintainence, the KBP realisability problem for strategies in Γ is NP-complete with respect to $|M|$ and PSPACE-complete w.r.t. $|Agt|$.*

Therefore, unlike most of the cases in which the relaxation of assumptions lead to higher computational complexity, the relaxation of completely-known environment comes for free and the relaxtion of rational agents lead to lower complexity. The usefulness of the new semantics can be seen not only on its conceptual suitability in modelling real-world systems but also on its practical implementability suggested by the complexity results.

We emphasise that the new semantics by Equation (4) and (5) can be implemented by re-using the existing tool, e.g., the symbolic BDD-based implementation [Huang and van der Meyden, 2014b], designed for the original semantics by Equation (1). For Equation (4), it can be done by encoding strategy $\theta^{\perp}_{M,i} \in \Gamma_i \times (\times_{j\in Agt, j\neq i}\theta_{\perp})$, instead of the strategy in $\times_{j\in Agt}\Gamma_j$, for every agent i and then following the same approach. For Equation (5), we encode strategy $\theta^{\perp}_{M^i,i}$ in $\Gamma_{E^i,i} \times (\times_{j\in Agt, j\neq i}\theta_{\perp})$. The long term behaviour by the operator AG may need to encode a path of length $|S|$, according to the relation $M, M^i, \theta_{M^i}, s \models \phi$. This can be done by the bounded approach [Clarke *et al.*, 1999] that gradually increases the length of the path until $|S|$.

Related Work

Knowledge-based programs [Fagin *et al.*, 1997] is a formalisation to specify multiagent systems in terms of agents taking actions based on their knowledge. The complexity of solving KBPs (i.e., KBP realisability problem) has been shown in [van der Meyden, 1996] for perfect recall agents and in [Vardi, 1996] for observational (or reactive) agents, respectively. Neither of them study the complexity with respect to the number of agents as we do in Theorem 1 (i.e., PSPACE-complete w.r.t. $|Agt|$).

Partially-known environment is a realistic assumption for many real-world applications, in particular robotics and multiagent systems, see e.g., [Stentz, 1994; Burgard *et al.*, 2005; Ayala *et al.*, 2013; Nie *et al.*, 2016], etc. Current research focuses on designing ad hoc algorithms to solve problems in which the concept of partially-known is defined with respect to specific requirements in different scenarios. On the contrary, in this paper, we formally define the concept in two general methods (i.e., passively maintained and actively maintained) and study the resulting change in the computational complexity of the KBP realisability problems.

In [Aminof *et al.*, 2015], a verification problem is studied on a multiagent system in which agents know they are in some environment from a set G of environments, but they do not know which one. The verification problem tests the existence of reconfiguration (basically, a plan) to complete a task, which is specified with a language MRTL. The problem is similar with the synthesis problem in distributed system [Pnueli and Rosner, 1990] and therefore has similar complexity. Unlike our memoryless strategies, the configurations can be history-dependent. More importantly, their approach does not employ explicit structure to maintain agents' awareness to the environment and are not based on the syntax and semantics of KBPs.

There are a rich literature on strategic reasoning by various logics such as ATL [Alur *et al.*, 2002], strategy logic [Chatterjee *et al.*, 2010], etc. Although the relation between ATL model checking and KBP realisability problem has been established in [Huang and van der Meyden, 2014a], our work is different: those logic frameworks work with completely-known environments, and moreover we do not work with the strategic modalities.

Conclusions

In this paper, we consider new semantics to the knowledge-based programs by relaxing two implicit assumptions on its original semantics: rational agents and completely-known environment. The new semantics are applied on robot swarms and robotic search-and-rescue. We also study the changes of computational complexity of the KBP realisability problem under the new semantics.

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.

[Aminof *et al.*, 2015] Benjamin Aminof, Aniello Murano, Sasha Rubin, and Florian Zuleger. Verification of asynchronous mobile-robots in partially-known environments. In *PRIMA 2015*, pages 185–200, 2015.

[Ayala *et al.*, 2013] A. I. Medina Ayala, S. B. Andersson, and C. Belta. Temporal logic motion planning in unknown environments. In *IROS 2013*, 2013.

[Burgard *et al.*, 2005] W. Burgard, M. Moors, C. Stachniss, and F.E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376 – 386, 2005.

[Chatterjee *et al.*, 2010] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. *Inf. Comput.*, 208(6):677–693, 2010.

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

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

[Fagin *et al.*, 1997] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. Knowledge-based programs. *Distributed Computing*, 10(4):199–225, 1997.

[GTAI, 2014] GTAI. Industrie 4.0 - smart manufacturing for the future. Technical report, Germany Trade and Invest, 2014.

[Halpern and Moses, 2007] Joseph Y. Halpern and Yoram Moses. Characterizing Solution Concepts in Games Using Knowledge-Based Programs. In *the 20nd International Joint Conference on Artificial Intelligence (IJCAI2007)*, pages 1300–1307, 2007.

[Halpern and Zuck, 1992] J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: Knowledge-based derivations and correctness proofs for a family of protocols. *J. ACM*, 39(3):449–478, 1992.

[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 synthesis for epistemic specifications with observational semantics. In *20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS2014)*, pages 455–469, 2014.

[Kouvaros and Lomuscio, 2015] Panagiotis Kouvaros and Alessio Lomuscio. Verifying emergent properties of swarms. In *IJCAI-2015*, pages 1083–1089, 2015.

[Nembrini, 2005] J. Nembrini. *minimalist Coherent Swarming of Wireless NeNetwork Autonomous Mobile Robots*. PhD thesis, University of the West of England, 2005.

[Nie *et al.*, 2016] Xinkun Nie, Lawson L. S. Wong, and Leslie Pack Kaelbling. Searching for physical objects in partially known environments. In *ICRA2016*, pages 5403–5410, 2016.

[NSTC, 2016] NSTC. Preparing for the future of artificial intelligence. Technical report, Executive Office of the President National Science and Technology Council Committee on Technology, 2016.

[Pnueli and Rosner, 1990] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *FOCS 1990*, page 746757, 1990.

[Stentz, 1994] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *ICRA 1994*, 1994.

[van der Meyden and Shilov, 1999] Ron van der Meyden and Nikolay V. Shilov. Model Checking Knowledge and Time in Systems with Perfect Recall. In *FSTTCS 1999*, pages 432–445, 1999.

[van der Meyden, 1996] Ron van der Meyden. Knowledge based programs: On the complexity of perfect recall in finite environments. In *TARK 1996*, pages 31–49, 1996.

[Vardi, 1996] Moshe Y. Vardi. Implementing knowledge-based programs. In *TARK 1996*, 1996.