A Temporal Description Logic for Resource-Bounded Rule-Based Context-Aware Agents

Abdur Rakib[1](✉), Hafiz Mahfooz Ul Haque[1], and Rokan Uddin Faruqui[2]

[1] School of Computer Science,
The University of Nottingham Malaysia Campus, Semenyih, Malaysia
{Abdur.Rakib,khyx2hma}@nottingham.edu.my
[2] Department of Computing and Software, McMaster University, Hamilton, Canada
faruqumr@mcmaster.ca

Abstract. We propose a logical framework for modelling and verifying context-aware multi-agent systems. We extend CTL^* with belief and communication modalities, and the resulting logic $\mathcal{L}_{\mathcal{OCRS}}$ allows us to describe a set of rule-based reasoning agents with bound on time, memory and communication. The set of rules which are used to model the systems is derived from OWL 2 RL ontologies. We provide an axiomatization of the logic and prove it is sound and complete. We show how Maude rewriting system can be used to encode and verify interesting properties of $\mathcal{L}_{\mathcal{OCRS}}$ models using existing model checking techniques.

Keywords: Modal logic · Context-aware · Multi-agent systems · Ontology · Model checking

1 Introduction

The vision of pervasive computing technology intends to provide invisible computing environments so that a user can utilize services at any time and everywhere [1]. Context-awareness is a key concept in pervasive computing. In context-aware pervasive computing every user may have several computing devices, where information can be collected by using tiny resource-bounded devices, such as, e.g., PDAs, smart phones, and wireless sensor nodes [2]. These systems interact with human users, they often exhibit complex adaptive behaviours, they are highly decentralised and can naturally be implemented as multi-agent systems. An agent is a piece of software that requires to be reactive, pro-active, and that is capable of autonomous action in its environment to meet its design objectives.

In the literature, various logical frameworks have been developed for modelling and verification of multi-agent systems [3]. However, such frameworks may not be very suitable to model context-aware applications. This is because, most of those existing frameworks consider propositional logic as a simple knowledge representation language which is often not suitable for modelling real life complex systems. For example, propositional logic cannot directly talk about properties of individuals or relations between individuals. Much research in pervasive computing has been focused on incorporation of context-awareness features into pervasive applications by adapting the semantic web technology (see e.g., [4–6]), where description logic (*DL*)-based ontology languages are often used for context representation and reasoning. *DL* is a decidable fragment of first order logic (*FOL*). In [6], it has been shown how context-aware systems can be modelled as resource-bounded rule-based systems using ontologies. In that paper, the resources required by the agents to solve a given problem were considered the time and communication bandwidth. But not the space requirements for reasoning. Since context-aware systems often run on resource limited devices, memory requirement is an important factor for their reasoning. In this paper, we propose a logical framework based on the earlier work of Alechina and colleagues [7–9], and the resulting logic $\mathcal{L}_{\mathcal{OCRS}}$ allows us to describe a set of ontology-driven rule-based reasoning agents with bound on time, memory, and communication. In addition to the incorporation of space (memory) requirements for reasoning in [7], $\mathcal{L}_{\mathcal{OCRS}}$ also uses first order Horn clause rules derived from OWL 2 RL ontologies. While the frameworks presented in [7,8] provide a useful basis for experimentation with both the logical representation and verification of heterogeneous agents, it has become clear that a more expressive logical language is required if these frameworks are to be used for real world context-aware agents. Though the logic developed by [9] is based on *FOL*, memory bounds have not been imposed in that framework. The proposed framework allows us to determine how much time (measured as rule-firing cycles) are required to generate certain contexts, how many messages must be exchanged among agents, and how much space (memory) is required for an agent for the reasoning. For verification, we show how we can encode a $\mathcal{L}_{\mathcal{OCRS}}$ model using the Maude LTL model checker [10] and verify interesting resource-bounded properties.

The remainder of the paper is organized as follows. In Sect. 2, we discuss how contexts are represented using OWL 2 RL and SWRL. In Sect. 3, we describe our model of communicating multi-agent context-aware systems. In Sect. 4, we develop logic $\mathcal{L}_{\mathcal{OCRS}}$, in Sect. 5 we present an example system and experimental results, and conclude in Sect. 6.

2 Context Modelling

We view context is any information that can be used to identify the status of an entity. An entity can be a person, a place, a physical or a computing object. This context is relevant to a user and application, and reflects the relationship among themselves [11]. A context can be formally defined as a *(subject, predicate, object)* triple that states a fact about the subject where — the subject is an entity in the environment, the object is a value or another entity, and the predicate is a relationship between the subject and object. According to [11], *"if a piece of information can be used to characterize the situation of a participant in an* *interaction, then that information is context"*. For example, we can represent a context "Fiona is the caregiver of Tracy" as *(Fiona, isCareGiverOf, Tracy)*. Here, the caregiver of a patient is dynamically identified based on the care status of the caregiver. For context modelling we use OWL 2 RL, a profile of the new standardization OWL 2, and based on pD^* [12] and the description logic program (DLP) [13]. We choose OWL 2 RL because it is more expressive than the RDFS and suitable for the design and development of rule-based systems. An OWL 2 RL ontology can be translated into a set of Horn clause rules based on [13]. Furthermore, we express more complex rule-based concepts using SWRL [14] which allow us to write rules using OWL concepts. In our framework, a context-aware system composed of a set of rule-based agents, and firing of rules that infer new facts may determine context changes and representing overall behaviour of the system.

For illustration, we construct an ontology-based context-aware model for a healthcare epilepsy scenario adapted from [15]. The scenario is based on the monitoring of epileptic patients to detect epileptic seizures. An epileptic alarm may activate several actions such as warning the patient about potential danger, informing patient's caregivers to take appropriate actions, and sending SMS messages to patient's relatives who are currently near to the patient.

"The goal of the epileptic patients' monitoring context-aware system is to detect the seizures, and to react in the following ways: (i) notify the epileptic patient of an upcoming seizure; and (ii) notify his/her nearby caregivers of an upcoming seizure of the patient by showing a map with the location of the patient. The caregivers who receive the notification for help should be (i) assigned as one of the caregivers of that particular patient; (ii) available for helping; and (iii) physically close to the patient. Upon a notification for help, caregivers may either accept or reject the request for helping the epileptic patient. When a particular caregiver accepts to help, the other caregivers who had received the notification for help are informed that a certain caregiver has already accepted to help that

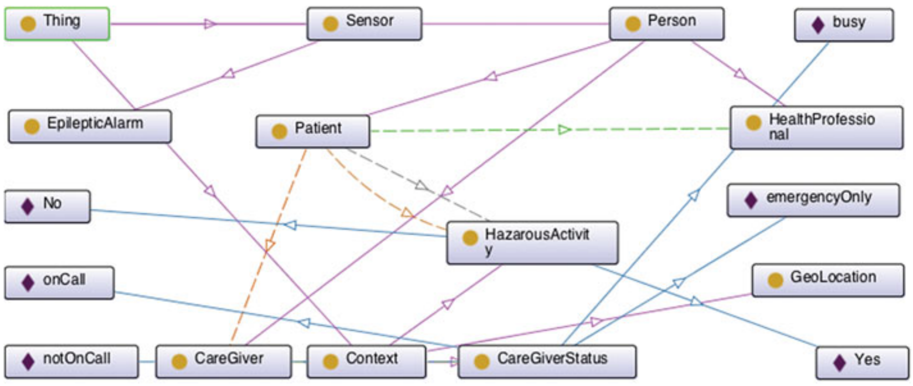
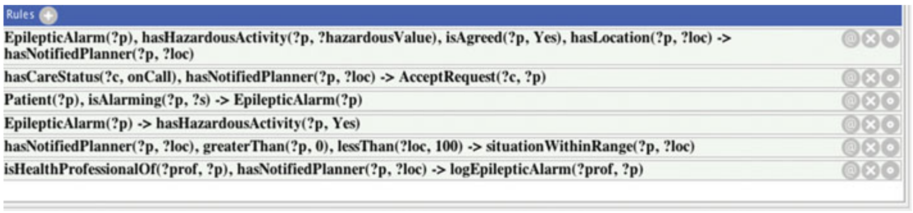


Fig. 1. A fragment of the epileptic patients' monitoring ontology


Rules

EpilepticAlarm(?p), hasHazardousActivity(?p, ?hazardousValue), isAgreed(?p, Yes), hasLocation(?p, ?loc) -> hasNotifiedPlanner(?p, ?loc)

hasCareStatus(?c, onCall), hasNotifiedPlanner(?p, ?loc) -> AcceptRequest(?c, ?p)

Patient(?p), isAlarming(?p, ?s) -> EpilepticAlarm(?p)

EpilepticAlarm(?p) -> hasHazardousActivity(?p, Yes)

hasNotifiedPlanner(?p, ?loc), greaterThan(?p, 0), lessThan(?loc, 100) -> situationWithinRange(?p, ?loc)

isHealthProfessionalOf(?prof, ?p), hasNotifiedPlanner(?p, ?loc) -> logEpilepticAlarm(?prof, ?p)

Fig. 2. Example SWRL rules

patient" [15]. Using Protégé [16], we build an OWL 2 RL ontology to capture the static behaviour of the system. A fragment of this ontology is depicted in Fig. 1. The dynamic aspect of the system is captured using SWRL rules. A snapshot of some SWRL rules is given in Fig. 2. In order to design a context-aware rule-based system from the above ontology, we extract Horn clause rules using the technique described in [6]. We show an example system encoding in Maude in Sect. 5 based on the logic developed in Sect. 4.

3 Context-Aware Agents

In our model a multi-agent context-aware system consists of n_{Ag} (≥ 1) individual *agents* $A_g = \{1, 2, \ldots, n_{Ag}\}$. Each agent $i \in A_g$ has a program, consisting of Horn clause rules of the form $P_1, P_2, \ldots, P_n \rightarrow P$ (derived from OWL 2 RL and SWRL), and a working memory, which contains ground atomic facts (contexts) taken from ABox representing the initial state of the system. In the rule, the antecedents $P_1, P_2, \ldots, P_n$ and the consequent P are context information. The antecedents of the rule form a complex context which is a conjunction of n contexts. In a resource-bounded system, it is quite unrealistic to presume that a single agent can acquire and understand available contextual information and infer new contexts alone. Thus sharing knowledge among agents is an efficient way to build context-aware systems. In our model, agents share a common ontology and communication mechanism. To model communication between agents, we assume that agents have two special communication primitives $\mathit{Ask}(i, j, P)$ and $\mathit{Tell}(i, j, P)$ in their language, where i and j are agents and P is an atomic context not containing an Ask or a Tell. $\mathit{Ask}(i, j, P)$ means 'i asks j whether the context P is the case' and $\mathit{Tell}(i, j, P)$ means 'i tells j that context P' ($i \neq j$). The positions in which the Ask and Tell primitives may appear in a rule depends on which agent's program the rule belongs to. Agent i may have an Ask or a Tell with arguments (i, j, P) in the consequent of a rule; e.g., $P_1, P_2, \ldots, P_n \rightarrow \mathit{Ask}(i, j, P)$ whereas agent j may have an Ask or a Tell with arguments (i, j, P) in the antecedent of the rule; e.g., $\mathit{Tell}(i, j, P) \rightarrow P$ is a well-formed rule (we call it trust rule) for agent j that causes it to believe i when i informs it that context P is the case. No other occurrences of Ask or Tell are allowed. When a rule has either an Ask or a Tell as its consequent, we call it a communication rule. All other rules are known as deduction rules. These include rules with *Ask*s and *Tell*s in the antecedent as well as rules containing neither an *Ask* nor a *Tell*. Note that OWL 2 is limited to unary and binary predicates and it is function-free. Therefore, in the Protégé editor all the arguments of *Ask* and *Tell* are represented using constant symbols and these annotated symbols are translated appropriately when designing the target system using the Maude specification.

4 Logic $\mathcal{L}_{\mathcal{OCRS}}$

A *DL* knowledge base (*KB*) has two components: the Terminology Box (*TBox*) $\mathcal{T}$ and the Assertion Box (*ABox*) $\mathcal{A}$. The *TBox* introduces the terminology of a domain, while the *ABox* contains assertions about individuals in terms of this vocabulary. The *TBox* is a finite set of general concept inclusions (*GCI*) and role inclusions. A *GCI* is of the form $C \sqsubseteq D$ where C, D are *DL*-concepts and a role inclusion is of the form $R \sqsubseteq S$ where R, S are *DL*-roles. We may use $C \equiv D$ (concept equivalence) as an abbreviation for the two *GCI*s $C \sqsubseteq D$ and $D \sqsubseteq C$ and $R \equiv S$ (role equivalence) as an abbreviation for $R \sqsubseteq S$ and $S \sqsubseteq R$. The *ABox* is a finite set of concept assertions in the form of $C(a)$ and role assertions in the form of $R(a, b)$.

Definition 1 (Interpretation of DL-knowledge bases). *An Interpretation of a DL knowledge base is a pair* $\mathcal{I} =< \Delta^{\mathcal{I}}, .^{\mathcal{I}} >$ *where* $\Delta^{\mathcal{I}}$ *is a non-empty set (the domain of interpretation) and* $.^{\mathcal{I}}$ *is a function that maps every concept to a subset of* $\Delta^{\mathcal{I}}$*, every role to a subset of* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$*, and each individual name to an element of the domain* $\Delta^{\mathcal{I}}$*.*

An interpretation $\mathcal{I}$ *satisfies the concept assertion* $C(a)$*, denoted by* $\mathcal{I} \models C(a)$*, iff* $a^{\mathcal{I}} \in C^{\mathcal{I}}$ *and it satisfies the role assertion* $R(a, b)$*, denoted by* $\mathcal{I} \models R(a, b)$*, iff* $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$*, where* a *and* b *are individuals.*

We now introduce the logic $\mathcal{L}_{\mathcal{OCRS}}$ which is an extension of the logic developed by [7]. Let us define the internal language of each agent in the system. Let the set of agents be $A_g = \{1, 2, ..., n_{Ag}\}$, $\mathcal{C} = \{C_1, C_2, \ldots C_n\}$ be a finite set of concepts, $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ be a finite set of roles, and $\mathcal{A}$ be a finite set of assertions. We also define a set $\mathcal{Q} = \{Ask(i, j, P),\ Tell(i, j, P)\}$, where $i, j \in A_g$ and $P \in \mathcal{C} \cup \mathcal{R}$. Note that $\mathcal{C}$ and $\mathcal{R}$ are the sets of concepts and roles that appear in $\mathcal{A}$. Let $\Re = \{r_1, r_2, \ldots, r_n\}$ be a finite set of rules of the form $P_1, P_2, \ldots, P_n \rightarrow P$, where $n \geq 0$, $P_i, P \in \mathcal{C} \cup \mathcal{R} \cup \mathcal{Q}$ for all $i \in \{1, 2, \ldots, n\}$ and $P_i \neq P_j$ for all $i \neq j$. For convenience, we use the notation $ant(r)$ for the set of antecedents of r and $cons(r)$ for the consequent of r, where $r \in \Re$. Let $g : \wp(\mathcal{A}) \rightarrow \Re$ be a substitution function that uses a forward-chaining strategy to instantiate the rule-base. We denote by $\mathcal{G}(\Re)$ the set of all the ground instances of the rules occurring in $\Re$, which is obtained using g. Thus $\mathcal{G}(\Re)$ is finite. Let $\bar{r} \in \mathcal{G}(\Re)$ be one of the possible instances of a rule $r \in \Re$. Note that $C(a)$, $R(a, b)$, $Ask(i, j, C(a))$, $Ask(i, j, R(a, b))$, $Tell(i, j, C(a))$, and $Tell(i, j, R(a, b))$ are ground facts, for all $C \in \mathcal{C}, R \in \mathcal{R}$. The internal language $\mathcal{L}$ includes all the ground facts and rules. Let us denote the set of all formulas by Ω which is finite.

In the modal language of $\mathcal{L}$ we have belief operator B_i for all $i \in A_g$. We assume that there is a bound on communication for each agent i which limits agent i to at most $n_C(i) \in \mathbb{Z}^*$ messages. Each agent has a communication counter, $cp_i^{=n}$, which starts at 0 ($cp_i^{=0}$) and is not allowed to exceed the value $n_C(i)$. We divide agent's memory into two parts as rule memory (knowledge base) and working memory. Rule memory holds set of rules, whereas the facts are stored in the agent's working memory. Working memory is divided into static memory ($S_M(i)$) and dynamic memory ($D_M(i)$). The $D_M(i)$ of each agent $i \in A_g$ is bounded in size by $n_M(i) \in \mathbb{Z}^*$, where one unit of memory corresponds to the ability to store an arbitrary formula. The static part contains initial information to start up the systems, e.g., initial working memory facts, thus its size is determined by the number of initial facts. The dynamic part contains newly derive facts as the system moves. Only formulas stored in $D_M(i)$ may get overwritten if it is full. Note that unless otherwise stated, in the rest of the paper we shall assume that memory means $D_M(i)$. For convenience, we define the following sets: $CP_i = \{cp_i^{=n} \mid n = \{0, \ldots, n_C(i)\}\}$, $CP = \bigcup_{i \in A_g} CP_i$.

The syntax of $\mathcal{L}_{\mathcal{OCRS}}$ includes the temporal operators of CTL^* and is defined inductively as follows:

- $\top$ (tautology) and *start* (a propositional variable which is only true at the initial moment of time) are well-formed formulas (wff) of $\mathcal{L}_{\mathcal{OCRS}}$;
- $cp_i^{=n}$ (which states that the value of agent i's communication counter is n) is a wff of $\mathcal{L}_{\mathcal{OCRS}}$ for all $n \in \{0, \ldots, n_C(i)\}$ and $i \in A_g$;
- $B_iC(a)$ (agent i believes $C(a)$), $B_iR(a,b)$ (agent i believes $R(a,b)$), and B_ir (agent i believes r) are wffs of $\mathcal{L}_{\mathcal{OCRS}}$ for any $C \in \mathcal{C}, R \in \mathcal{R}$, $r \in \Re$ and $i \in A_g$;
- $B_kAsk(i,j,C(a))$, $B_kAsk(i,j,R(a,b))$, $B_kTell(i,j,C(a))$, and $B_kTell(i,j, R(a,b))$ are wffs of $\mathcal{L}_{\mathcal{OCRS}}$ for any $C \in \mathcal{C}, R \in \mathcal{R}$, $i,j \in A_g$, $k \in \{i,j\}$, and $i \neq j$;
- If φ and ψ are wffs of $\mathcal{L}_{\mathcal{OCRS}}$, then so are $\neg\varphi$ and $\varphi \wedge \psi$;
- If φ and ψ are wffs of $\mathcal{L}_{\mathcal{OCRS}}$, then so are $X\varphi$ (in the next state φ), $\varphi U \psi$ (φ holds until ψ), $A\varphi$ (on all paths φ).

Other classical abbreviations for $\bot$, $\vee$, $\rightarrow$ and $\leftrightarrow$, and temporal operations: $F\varphi \equiv \top U \varphi$ (at some point in the future φ) and $G\varphi \equiv \neg F \neg \varphi$ (at all points in the future φ), and $E\varphi \equiv \neg A \neg \varphi$ (on some path φ) are defined as usual.

The semantics of $\mathcal{L}_{\mathcal{OCRS}}$ is defined by $\mathcal{L}_{\mathcal{OCRS}}$ transition systems which are based on ω-tree structures. Let (S,T) be a pair where S is a set and T is a binary relation on S that is total, i.e., $\forall s \in S \cdot \exists s' \in S \cdot sTs'$. A branch of (S,T) is an ω-sequence $(s_0, s_1, \ldots)$ such that s_0 is the root and s_iTs_{i+1} for all $i \geq 0$. We denote $B(S,T)$ to be the set of all branches of (S,T). For a branch $\pi \in B(S,T)$, π_i denotes the element s_i of π and $\pi_{\leq i}$ is the prefix $(s_0, s_1, \ldots, s_i)$ of π. A $\mathcal{L}_{\mathcal{OCRS}}$ transition system $\mathbb{M}$ is defined as $\mathbb{M} = (S,T,V)$ where

- (S,T) is a ω-tree frame
- $V : S \times A_g \rightarrow \wp(\Omega \cup CP)$; we define the belief part of the assignment $V^B(s,i) = V(s,i) \setminus CP$ and the communication counter part $V^C(s,i) = V(s,i) \cap CP$. We further define $V^M(s,i) = \{\alpha | \alpha \in D_M(i)\}$. V satisfies the following conditions:

1. $|V^C(s,i)| = 1$ for all $s \in S$ and $i \in A_g$.
2. If $(s,t) \in T$ and $cp_i^{=n} \in V(s,i)$ and $cp_i^{=m} \in V(t,i)$ then $n \leq m$.

– we say that a rule $r : P_1, P_2, \ldots, P_n \rightarrow P$ is applicable in a state s of an agent i if $ant(\bar{r}) \in V(s,i)$ and $cons(\bar{r}) \notin V(s,i)$. The following conditions on the assignments $V(s,i)$, for all $i \in A_g$, and transition relation T hold in all models:
  1. for all $i \in A_g$, $s, s' \in S$, and $r \in \Re$, $r \in V(s,i)$ iff $r \in V(s',i)$. This describes that agent's program does not change.
  2. for all $s, s' \in S$, sTs' holds iff for all $i \in A_g, V(s',i) = V(s,i) \cup \{cons(\bar{r})\} \cup \{Ask(j,i,C(a))\} \cup \{Tell(j,i,C(a)\} \cup \{Ask(j,i,R(a,b))\} \cup \{Tell(j,i,R(a,b)\}$. This describes that each agent i fires a single applicable rule instance of a rule r, or updates its state by interacting with other agents, otherwise its state does not change.

The truth of a $\mathcal{L}_{\mathcal{OCRS}}$ formula at a point n of a path $\pi \in B(S,T)$ is defined inductively as follows:

– $\mathbb{M}, \pi, n \models \top$,
– $\mathbb{M}, \pi, n \models start$ iff $n = 0$,
– $\mathbb{M}, \pi, n \models B_i\alpha$ iff $\alpha \in V(s,i)$,
– $\mathbb{M}, \pi, n \models cp_i^{=m}$ iff $cp_i^{=m} \in V(s,i)$,
– $\mathbb{M}, \pi, n \models \neg\varphi$ iff $\mathbb{M}, \pi, n \not\models \varphi$,
– $\mathbb{M}, \pi, n \models \varphi \sqcap \psi$ iff $\mathbb{M}, \pi, n \models \varphi$ and $\mathbb{M}, \pi, n \models \psi$,
– $\mathbb{M}, \pi, n \models X\varphi$ iff $\mathbb{M}, \pi, n+1 \models \varphi$,
– $\mathbb{M}, \pi, n \models \varphi U \psi$ iff $\exists m \geq n$ such that $\forall k \in [n,m)$ $\mathbb{M}, \pi, k \models \varphi$ and $\mathbb{M}, \pi, m \models \psi$,
– $\mathbb{M}, \pi, n \models A\varphi$ iff $\forall \pi' \in B(S,T)$ such that $\pi'_{\leq n} = \pi_{\leq n}$, $\mathbb{M}, \pi', n \models \varphi$.

We now describe conditions on the models. The transition relation T corresponds to the agent's executing actions $\langle act_i, act_2, \ldots, act_{n_{Ag}} \rangle$ where act_i is a possible action of an agent i in a given state s. The set of actions that each agent i can perform are: $Rule_{i,\bar{r},\beta}$ (agent i firing a rule instance $\bar{r}$ and adding $cons(\bar{r})$ to its working memory and removing β), $Copy_{i,\alpha,\beta}$ (agent i copying α from other agent's memory and removing β, where α is of the form $Ask(j,i,P)$ or $Tell(j,i,P)$, and $Idle_i$ (agent i does nothing but moves to the next state). Intuitively, β is an arbitrary facts which gets overwritten if it is in the agent's dynamic memory $D_M(i)$. If agent's memory is full $|V^M(s,i)| = n_M(i)$ then we require that β has to be in $V^M(s,i)$. Not all actions are possible in a given state. For example, there may not be any matching rule instances. When the counter value reaches to $n_C(i)$, i cannot perform copy action any more. Let us denote the set of all possible actions by agent i in a given state s by $T_i(s)$ and its definition is given below:

Definition 2 (Available actions). *For every state s and agent i,*

1. *$Rule_{i,r,\beta} \in T_i(s)$ iff $r \in V(s,i)$, $ant(\bar{r}) \subseteq V(s,i)$, $cons(\bar{r}) \notin V(s,i)$, $\beta \in \Omega$ or if $|V^M(s,i)| = n_M(i)$ then $\beta \in V^M(s,i)$;*
2. *$Copy_{i,\alpha,\beta} \in T_i(s)$ iff there exists $j \neq i$ such that $\alpha \in V(s,j)$, $\alpha \notin V(s,i)$, $cp_i^{=n} \in V(s,i)$ for some $n < n_C(i)$, α is of the form $Ask(j,i,P)$ or $Tell(j,i,P)$, and β as before;*
3. *$Idle_i$ is always in $T_i(s)$.*

Definition 3 (Effect of actions). *For each $i \in A_g$, the result of performing an action act_i in state s is defined if $act_i \in T_i(s)$ and has the following effect on the assignment of formulas to i in the successor state s':*

1. *if act_i is $Rule_{i,r,\beta}$: $V(s', i) = V(s, i) \setminus \{\beta\} \cup \{cons(\bar{r})\}$;*
2. *if act_i is $Copy_{i,\alpha,\beta}, cp_i^{=n} \in V(s, i)$ for some $n \leq n_C(i)$: $V(s', i) = V(s, i) \setminus \{\beta, cp_i^{=n}\} \cup \{\alpha, cp_i^{=n+1}\}$;*
3. *if act_i is $Idle_i$: $V(s', i) = V(s, i)$.*

Now, the definition of the set of models corresponding to a system of rule-based reasoners is given below:

Definition 4. *$\mathbb{M}(n_M, n_C)$ is the set of models (S, T, V) which satisfies the following conditions:*

1. *$cp_i^{=0} \in V(s_0, i)$ where $s_0 \in S$ is the root of (S, T), $\forall i \in A_g$;*
2. *$\forall s, s' \in S$, sTs' iff for some tuple of actions $\langle act_i, act_2, \ldots, act_{n_{Ag}} \rangle$, $act_i \in T_i(s)$ and the assignment in s' satisfies the effects of act_i, $\forall i \in A_g$;*
3. *$\forall s \in S$ and a tuple of actions $\langle act_i, act_2, \ldots, act_{n_{Ag}} \rangle$, if $act_i \in T_i(s), \forall i \in A_g$, then $\exists s' \in S$ s.t. sTs' and s' satisfies the effects of act_i, $\forall i \in A_g$;*
4. *The bound on each agent's memory is set by the following constraint on the mapping V: $|V^M(s, i)| \leq n_M(i)$,$\forall s \in S$,$i \in A_g$.*

Note that the bound $n_C(i)$ on each agent i's communication ability (no branch contains more than $n_C(i)$ *Copy* actions by agent i) follows from the fact that $Copy_i$ is only enabled if i has performed fewer than $n_C(i)$ copy actions in the past. Below are some abbreviations which will be used in the axiomatization:

- $ByRule_i(P, n) = \neg B_i P \wedge cp_i^{=n} \wedge \bigvee_{r \in \Re \wedge cons(\bar{r})) = P} (B_i r \wedge \bigwedge_{Q \in ant(\bar{r})} B_i Q)$. This formula describes the state before the agent comes to believe formula P by the *Rule* transition, n is the value of i's communication counter, P and Q are ground atomic formulas.
- $ByCopy_i(\alpha, n) = \neg B_i \alpha \wedge B_j \alpha \wedge cp_i^{=n-1}$, where α is of the form $Ask(j, i, P)$ or $Tell(j, i, P)$, $i, j \in A_g$ and $i \neq j$.

Now we introduce the axiomatization system.

A1 All axioms and inference rules of CTL^* [17].

A2 $\bigwedge\limits_{\alpha \in D_M(i)} B_i \alpha \rightarrow \neg B_i \beta$ for all $D_M(i) \subseteq \Omega$ such that $|D_M(i)| = n_M(i)$ and $\beta \notin D_M(i)$. This axiom describes that, in a given state, each agent can store maximally at most $n_M(i)$ formulas in its memory,

A3 $\bigvee\limits_{n=0,\ldots,n_C(i)} cp_i^{=n}$,

A4 $cp_i^{=n} \rightarrow \neg cp_i^{=m}$ for any $m \neq n$,

A5 $B_i r \wedge \bigwedge_{P \in ant(\bar{r})} B_i P \wedge cp_i^{=n} \wedge \neg B_i cons(\bar{r}) \rightarrow EX(B_i cons(\bar{r}) \wedge cp_i^{=n})$, $i \in A_g$. This axiom describes that if a rule matches, its consequent belongs to some successor state.

A6 $cp_i^{=n} \wedge \neg B_i\alpha \wedge B_j\alpha \rightarrow EX(B_i\alpha \wedge cp_i^{=n+1})$ where α is of the form $Ask(j,i,P)$ or $Tell(j,i,P)$, $i,j \in A_g$, $j \neq i$, $n < n_C(i)$. This axiom describes transitions made by *Copy* with communication counter increased.

A7 $EX(B_i\alpha \wedge B_i\beta) \rightarrow B_i\alpha \vee B_i\beta$, where α and β are not of the form $Ask(j,i,P)$ and $Tell(j,i,P)$. This axiom says that at most one new belief is added in the next state.

A8 $EX(B_i\alpha \wedge cp_i^{=n}) \rightarrow B_i\alpha \vee ByRule_i(\alpha,n) \vee ByCopy_i(\alpha,n)$ for any $\alpha \in \cup\Omega$. This axiom says that a new belief can only be added by one of the valid reasoning actions.

A9a $start \rightarrow cp_i^{=0}$ for all $i \in A_g$. At the start state, the agent has not performed any *Copy* actions.

A9b $\neg EX\ start$. *start* holds only at the root of the tree.

A10 B_ir where $r \in \Re$ and $i \in A_g$. This axiom tells agent i believes its rules.

A11 $\neg B_ir$ where $r \notin \Re$ and $i \in A_g$. This axiom tells agent i only believes its rules.

A12 $\varphi \rightarrow EX\varphi$, where φ does not contain *start*. This axiom describes an *Idle* transition by all the agents.

A13 $\bigwedge_{i\in A_g} EX(\bigwedge_{\alpha\in\Gamma_i} B_i\alpha \wedge cp_i^{=n_i}) \rightarrow EX \bigwedge_{i\in A_g}(\bigwedge_{\alpha\in\Gamma_i} B_i\alpha \wedge cp_i^{=n_i})$ for any $\Gamma_i \subseteq \Omega$. This axiom describes that if each agent i can separately reach a state where it believes formulas in Γ_i, then all agents together can reach a state where for each i, agent i believes formulas in Γ_i.

Let us now define the logic obtained from the above axiomatisation system.

Definition 5. $\mathbb{L}(n_M, n_C)$ *is the logic defined by the axiomatisation* ***A1*** *-* ***A13****.*

Theorem 1. $\mathbb{L}(n_M, n_C)$ *is sound and complete with respect to* $\mathbb{M}(n_M, n_C)$*.*

Sketch of Proof. The proof of soundness is standard. The proofs for axioms and rules included in **A1** are given in [17]. Axiom **A2** assures that at a state, each agent can store maximally at most $n_M(i)$ formulas in its memory. Axioms **A3** and **A4** force the presence of a unique counter for each agent to record the number of copies it has performed so far. In particular, **A3** makes sure that at least a counter is available for any agent and **A4** guaranties that only one of them is present. In the following, we provide the proof for **A5**. The proofs for other axioms are similar.

Let $\mathbb{M} = (S,T,V) \in \mathbb{M}(n_M, n_C)$, $\pi \in B(S,T)$ and $n \geq 0$. We assume that $\mathbb{M}, \pi, n \models B_ir \wedge \bigwedge_{P\in ant(\bar{r})} B_iP \wedge cp_i^{=m} \wedge \neg B_i cons(\bar{r})$, for some $r \in \Re$, and $|V^M(s,i)| \leq n_M(i)$. Then $P \in V(\pi_n, i)$ for all $P \in ant(\bar{r})$, and $cons(\bar{r}) \notin V(\pi_n, i)$. This means that the action performed by i is $Rule_{i,r,\beta}$. According to the definition of $\mathbb{M}(n_M, n_C)$, $\exists s' \in S \cdot \pi_n T s'$ and $V(s',i) = V(\pi_n,i) \setminus \{\beta\} \cup \{cons(\bar{r})\}$. Let π' be a branch in $B(S,T)$ such that $\pi'_{\leq n} = \pi_{\leq n}$ and $\pi'_{n+1} = s'$. Then we have $\mathbb{M}, \pi', n+1 \models B_i cons(\bar{r}) \wedge cp_i^{=m}$. Therefore, it is obvious that $\mathbb{M}, \pi, n \models EX(B_i cons(\bar{r}) \wedge cp_i^{=m})$.

Completeness can be shown by constructing a tree model for a consistent formula φ. This is constructed as in the completeness proof introduced in [17]. Then we use the axioms to show that this model is in $\mathbb{M}(n_M, n_C)$. Due to space limitations we omit the proof of this result. □

5 Maude Encoding

We build a multi-agent rule-based system whose rules are derived from the ontology of the healthcare epilepsy scenario described in Sect. 2. The system consists of four agents: Patient (1), Planner (2), CareGiver (3), and HealthProfessional (4). The set of rules and initial working memory facts that are distributed to the agents are shown in Table 1. For the specification and verification of the system we use Maude LTL model checker. The choice of LTL is not essential, it is straightforward to encode a $\mathcal{L}_{\mathcal{OCRS}}$ model for a standard model checker. We use LTL because it is the logic supported by the Maude system used in our case study. We chose the Maude LTL model checker because it can model check systems whose states involve arbitrary algebraic data types. The only assumption is that the set of states reachable from a given initial state is finite. Rule variables can be represented directly in the Maude encoding, without having to generate all ground instances resulting from possible variable substitutions.

Due to space limitation we omit the encoding here, however, it is similar to [6], apart from the implementation of agents memory bounds. We verified a number of interesting resource-bounded properties of the system including the following:

Table 1. Horn-Clause rules for the epileptic patients' monitoring context-aware system

Patient’s rule
Initial facts: Patient(’Tracy), isAlarming(’Tracy, ’Beep), hasGeolocation(’Tracy, ’DownTown)
Patient(?p),isAlarming(?p,?s) → EpilepticAlarm(?p)
EpilepticAlarm(?p) → hasHazardousActivity(?p, ’Yes)
hasHazardousActivity(?p, ’Yes) → isAgreed(?p,’Yes)
hasHazardousActivity(?p, ’Yes) → isAgreed(?p,’No)
EpilepticAlarm(?p), hasHazardousActivity(?p, ’Yes), hasGeoLocation(?p, ?location) → hasNotifiedPatient(?p, ?location)
EpilepticAlarm(?p), hasHazardousActivity(?p, ’Yes), hasGeoLocation(?p, ?location), isAgreed(?p, ’Yes) →Tell(1, 2, hasNotifiedPlanner(?p,?location))
Planner’s rules
Initial facts: isCareGiverOf(’Fiona,’Tracy), isCareGiverOnCall(’Fiona, ’OnCall)
Tell(1, 2, hasNotifiedPlanner(?p,?location)) → hasNotifiedPlanner(?p,?location)
hasNotifiedPlanner(?p,?location),lessThan(?location,’30), greaterThan(?location,0)→ situationWithinRange(?p,?location)
situationWithinRange(?p,?location),isCareGiverOf(?c,?p),isCareGiverOnCall(?c,?stat) → Ask(2,3, hasCareStatus(?c, ’stat))
Tell(3,2,hasCarStatus(?c,’onCall))→ hasCarStatus(?c,’onCall) hasCarStatus(?c,’onCall), hasNotifiedPlanner(?p,?location) → AcceptRequest(?c, ?p)
Tell(3,2,hasCarStatus(?c, ’Busy))→ hasCareStatus(?c, ’Busy)
hasNotifiedPlanner(?p,?location) → Tell(2, 5, hasNotifiedPlanner(?p,?location))
CareGiver’s rules
Initial facts:
Ask(2,3, hasCareStatus(?c, ?stat)) → hasCareStatus(?c, ?stat)
hasCareStatus(?c, ?stat) → Tell(3,2,hasCarStatus(?c, ’OnCall))
hasCareStatus(?c, ?stat) → Tell(3,2,hasCarStatus(?c, ’Busy))
hasCareStatus(?c, ?stat) → Tell(3,2,hasCarStatus(?c, ’NotOnCall))
hasCareStatus(?c, ?stat) → Tell(3,2,hasCarStatus(?c, ’EmergencyOnly))
HealthProfessional’s rules
Initial facts: isHealthProfesional(’John, ’Tracy)
Tell(2, 5, hasNotifiedPlanner(?p,?location)) → hasNotifiedPlanner(?p,?location)
hasNotifiedPlanner(?p,?location), isHealthProfesional(?prof, ?p) → logEpilepticAlarm(?prof,?p)

$$\begin{aligned}&G(B_1\, \mathit{EpilepticAlarm}('\mathit{Tracy})\\&\rightarrow X^n B_1\, \mathit{Tell}(1, 2, \mathit{hasNotifiedPlanner}('\mathit{Tracy},' \mathit{DownTown}))\\&\qquad \wedge (\mathit{msg}_1{}^{=m}) \wedge (n_M(1) \geq l))\end{aligned}$$

the above property specifies that whenever there is an epileptic alarm for Tracy, agent 1 notifying agent 2 that "Tracy" has hazardous activity and she is located in "DownTown" within n time steps, while exchanging m messages and space requirement for agent 1 is at least l units, and

$$\begin{aligned}&G(B_2\, \mathit{Tell}(1, 2, \mathit{hasNotifiedPlanner}('\mathit{Tracy},' \mathit{DownTown}))\\&\rightarrow X^n B_2\, \mathit{AcceptRequest}('\mathit{Fiona},' \mathit{Tracy}) \wedge (\mathit{msg}_2{}^{=m}) \wedge (n_M(2) \geq l))\end{aligned}$$

which specifies that whenever agent 2 gets notified that "Tracy" has hazardous activity and she is located in "DownTown" it believes that care giver Fiona accepts the request within n time steps, while exchanging m messages and space requirement for agent 2 is at least l units.

The above properties are verified as true when the values of n, m, and l are 3, 1, and 3 in the first property, and the values of n, m, and l are 9, 3, and 2 in the second property. However, the properties are verified as false and the model checker returns counterexamples when we assign a values to n, m, and l which are less than 3, 1, and 3 in the first property, and values to n, m, and l which are less than 9, 3 and 2 in the second property.

6 Conclusions and Future Work

In this paper, we presented a formal logical framework for modelling and verifying context-aware multi-agent systems. Where agents reason using ontology-driven first order Horn clause rules. We considered space requirement for reasoning in addition to the time and communication resources. We modelled an ontology-based context-aware system to show how we can encode a $\mathcal{L}_{\mathcal{OCRS}}$ model using Maude LTL model checker and formally verify its resource-bounded properties. In future work, we would like to develop a framework that will allow us to design context-aware system automatically from a given scenario described in natural languages. This requires extracting specification to build its corresponding ontology for the desired system.

References

1. Weiser, M.: The computer for the 21st century. ACM SIGMOBILE Mob. Comput. Commun. Rev. (Special Issue Dedicated to Mark Weiser) **3**(3), 3–11 (1999)
2. Viterbo Filho, J., da Gama Malcher, M., Endler, M.: Supporting the development of context-aware agent-based systems for mobile networks. In: Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 1872–1873. ACM (2008)
3. Rakib, A.: Formal approaches to modelling and verifying resource-bounded agents-state of the art and future prospects. Inform. Tech. Softw. Eng. **2**(4) (2012)
4. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using OWL. In: PerCom Workshops 2004, pp. 18–22 (2004)

5. Esposito, A., Tarricone, L., Zappatore, M., Catarinucci, L., Colella, R., DiBari, A.: A framework for context-aware home-health monitoring. In: Sandnes, F.E., Zhang, Y., Rong, C., Yang, L.T., Ma, J. (eds.) UIC 2008. LNCS, vol. 5061, pp. 119–130. Springer, Heidelberg (2008)
6. Rakib, A., Faruqui, R.U.: A formal approach to modelling and verifying resource-bounded context-aware agents. In: Vinh, P.C., Hung, N.M., Tung, N.T., Suzuki, J. (eds.) ICCASA 2012. LNICST, vol. 109, pp. 86–96. Springer, Heidelberg (2013)
7. Alechina, N., Logan, B., Nga, N.H., Rakib, A.: Verifying time and communication costs of rule-based reasoners. In: Peled, D.A., Wooldridge, M.J. (eds.) MoChArt 2008. LNCS, vol. 5348, pp. 1–14. Springer, Heidelberg (2009)
8. Alechina, N., Logan, B., Nga, N.H., Rakib, A.: Verifying time, memory and communication bounds in systems of reasoning agents. Synthese **169**(2), 385–403 (2009)
9. Alechina, N., Jago, M., Logan, B.: Modal logics for communicating rule-based agents. In: Proceedings of the 17th European Conference on Artificial Intelligence, pp. 322–326 (2006)
10. Eker, S., Meseguer, J., Sridharanarayanan, A.: The Maude LTL model checker and its implementation. In: Ball, T., Rajamani, S.K. (eds.) SPIN 2003. LNCS, vol. 2648, pp. 230–234. Springer, Heidelberg (2003)
11. Dey, A., Abwowd, G.: Towards a better understanding of context and context-awareness. Technical report GIT-GVU-99-22, Georgia Institute of Technology
12. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. Web Semant. Sci. Serv. Agents World Wide Web **3**(2–3), 79–115 (2005)
13. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: WWW 2003, pp. 48–57. ACM Press (2003)
14. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: a Semantic Web Rule Language combining OWL and RuleML. Acknowledged W3C submission, standards proposal research report: Version 0.6 (April 2004)
15. Costa, P.D.: Architectural support for context-aware applications: from context models to services platforms. Ph.D. thesis, Centre for Telematics and Information Technology, University of Twente, AE Enschede, The Netherlands (2007)
16. Protégé: The Protégé ontology editor and knowledge-base framework (Version 4.1). http://protege.stanford.edu/ (July 2011)
17. Reynolds, M.: An axiomatization of full computation tree logic. J. Symb. Log. **66**(3), 1011–1057 (2001)