Noname manuscript No.
(will be inserted by the editor)

A Logical Framework for the Representation and Verification of Context-aware Agents

Abdur Rakib · Hafiz Mahfooz Ul Haque

Received: date / Accepted: date

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 bounds on time, memory and communication. The set of rules which are used to model a desired system 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 [28]. 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, including e.g., PDAs, smart phones, and wireless sensor nodes [26]. 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. An agent is autonomous if it encapsulates its behaviour and internal state. This means that an agent itself has control over its own actions and behaviour. When a system is composed of multiple interacting agents it is called a multi-agent system (MAS) [29]. In a MAS, agents are typically communicate via message passing and co-operate with other agents in order to achieve common goals. In many circumstances building centralized systems are quite impractical or undesirable such as pervasive systems, where MAS technology appears to be a primary choice in producing distributed information systems. In this paper, we address software reasoning agents which are capable of reasoning about their behaviour and interactions, however, an agent could be for example a robot. That is our agents are primarily viewed as doing some kind of inference over a knowledge base (KB), e.g., using forward chaining rules.

There has been considerable work in pervasive computing literature focusing on various domains including health care (see e.g., [8,19,2,12]). Much of this work concentrate on representing and reasoning about contexts. However, unlike many other context-aware application systems, in many cases health care systems are considered as safety critical systems [7]. In such systems, not meeting design objectives may result in tremendous loss including possibly human lives. For example, in a non-time critical environment, where small delays due to response time are not an issue, a system may respond to queries without any concern

A preliminary version of this paper appeared in the proceedings of the 2nd International Conference on Context-Aware Systems and Applications (ICCASA'13).

This work is partially supported by the Faculty of Science Pump Priming & Equipment Research Fund at the University of Nottingham Malaysia Campus.

Abdur Rakib
School of Computer Science, The University of Nottingham, Malaysia Campus
Tel.: +60389248137, Fax: +60389248018
E-mail: Abdur.Rakib@nottingham.edu.my

Hafiz Mahfooz Ul Haque
School of Computer Science, The University of Nottingham, Malaysia Campus
E-mail: khyx2hma@nottingham.edu.my

or consideration of the time required for reasoning. However, in many situations the time taken to do the reasoning is of critical importance. Other issues include space requirements for reasoning and the number of messages that are exchanged between tiny resource-bounded devices in order to achieve their goals. This is because memory space of such a device is relatively small and its life time is inversely proportional to the number of messages it exchanges. Systematic, formal approaches to their specification and verification can allow addressing these problems. In this research work, intended systems will not only be able to facilitate knowing *who* is encountering *what* problem *when* and *where*, but also if a system finds that someone encountering some problem in a particular place at a particular moment then whether assistance seeker will receive favoured services or not, if receives then what computational (time and space) and communication resources must be devoted to its solution by each agent.

In the literature, various logical frameworks have been developed for modelling and verification of multi-agent systems (a brief state-of-the-art survey can be found in [22, 21]). 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.,[27, 12, 23]), 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 [23], 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. In this paper, we propose a logical framework based on the earlier work of Alechina and colleagues [4, 5, 3], and the resulting $\mathcal{L}_{\mathcal{OCRS}}$ logic allows us to describe a set of ontology-driven rule-based reasoning agents with bounds on time, memory, and communication. In addition to the incorporation of space (memory) requirements for reasoning in [4], $\mathcal{L}_{\mathcal{OCRS}}$ also uses first order Horn clause rules derived from OWL 2 RL ontologies. We prove that the logic is sound and complete. While the frameworks presented in [4, 5] 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 [3] 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 to do the reasoning. For verification, we show how we can encode a $\mathcal{L}_{\mathcal{OCRS}}$ model using the Maude LTL model checker [11] and verify its certain interesting resource-bounded properties.

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

2 Semantic context model

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 [10]. 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 [10], *"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 contexts *"Mary has Systolic Blood Pressure 120"* as *(Mary, hasSystolicBloodPressure, 120)* and *"Mary has a carer named Fiona"* as *(Mary, hasCarer, Fiona)*. Here, the caregiver of a patient is dynamically identified based on the care status of the caregiver.

Over the last decade, significant research attention has been devoted to explore the various relationships between ontology and knowledge representation. In artificial intelligence (AI) the term ontology has been used to specify a conceptualization in the context of knowledge sharing. In [14], Gruber defines conceptualization as an abstract, simplified view of the environment we want to represent. More formally, an ontology can represent a model of a domain of discourse that introduces a vocabulary to specify the concepts relevant to the domain and their relationships. That is, an ontology can be used to represent knowledge of a domain which gives a clear and coherent view of that domain, and it can be seen as playing a key role in describing the semantics of the data. Suppose we want to describe a domain named smart home; we need to consider important parts including for example, *possible locations*, *furniture equipment*, *household equipment*, *technical equipment*, and *physical devices* with which people can interact. By observing the scenario, some questions can be raised into the system designer's mind, for example, who are the dwellers? how many rooms are there in the home? what is the location of TV? which devices are located at what places and how they interact with each other? and so on. Based on these kind of questions we can identify the major concepts those can be used in building the ontology. For instance, *"bedroom is a room that is part of the home"* is a concept that has terms with relevant semantics, this concept can be formalized using first order logic as follows:

$$\forall x \cdot [Bedroom(x) \rightarrow Room(x) \wedge \exists y \cdot [isPartOf(x,y) \wedge Home(x)]]$$

The Web Ontology Language OWL built on RDF and RDFS is a semantic markup language for ontologies that provides a formal syntax and semantics for them [9]. OWL ontology is essentially a set of axioms and consists of classes, individuals and properties. In the aforementioned *(subject, predicate, object)* triplet, subjects and objects are denoted by classes (and subclasses), while predicates are typically denoted by properties. Thus contexts can be efficiently modelled using ontologies. There are two types of properties in OWL ontologies, *object properties* and *data properties*. Object properties are binary relations which link an individual to an individual, whereas data properties link an individual to a typed data. Description logic based OWL is a good candidate for defining ontologies where automated reasoning is required. A reasoner can infer implicit facts contained in the ontology. As an example, a simple ontology about home is represented in DL 1-3.

$$Bedroom \sqsubseteq Room \sqcap \exists isPartOf \cdot Home \quad (1)$$

$$Bedroom \sqcap \exists hasDoor \cdot Door \quad (2)$$

$$Dweller(Mary) \quad (3)$$

The first two axioms define the concepts *"Bedroom is a room that is part of the home"* and *"Bedroom has a door"*, respectively, whereas the third axiom asserts the fact that *"Mary is a dweller'*. A reasoner can infer that *"Room has a door"* which is implicit in the ontology.

For context modelling we use OWL 2 RL, a profile of the new standardization OWL 2, and based on pD^* [17] 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 [16] 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.

The following subsections show how we model contexts for smart spaces and perform context-awareness reasoning.

2.1 Ontology-design

There are several different approaches in designing ontologies for a domain of interest [25]. One way to do this is to use bottom-up approach, in which ontology for smaller parts are constructed first, then using high-level abstract classes the desired ontology is developed. That is, bottom-up approach starts with the leaves of the hierarchy that defines the most specific classes first, and subsequently groups these classes into more general concepts. For example, we can start by defining classes for *Nurse* and *Physician*, then create a common superclass for these two classes as *Formal* in turn is a subclass of *CareGiver* and so on.

In contrast to the bottom-up approach, a top-down approach designs the upper classes first and then develops the small parts of the hierarchy. That is, top-down approach starts with defining the most general concepts in the domain first, and subsequently specializes the concepts down to the hierarchy.

In designing our smart space ontology, we adopt a top-level shared conceptualization and on top of which lower-level (domain specific) ontologies are built. The top-level ontology defines the high-level concepts that are common among different context-aware entities, independently from the application domain. Whereas the domain ontology refers to a specific domain defining the details of general concepts and their relationships. We use top-level concepts such as *Location*, *Person*, *Device*, *Service*, *Activity*, *Time*, and *Medication*. These conceptual entities specialize different concepts depending on the context sub-domain. For example, the *Person* context class defines the general feature of a person. In our home health-care modelling domain, it may be divided into *Patient* and *CareGiver* subclasses. The *CareGiver* class is further divided into *Formal* and *Informal* subclasses and so on. Our ontology-based context modelling provides physical representation of smart home (including its doors, windows, various locations, and so on), objects available in the smart home (including sensors and electric appliances), and users and their various activities and characteristics. It also provides (local and remote) services including operating electrical appliances, medical consultation, emergency response etc. A fragment of the ontology is depicted in Figure 1.

For brevity, we omit a detailed description of all the concepts and their relationships used to design the ontology for smart space considering both home and health centre environments. However, some of the classes of the patient context ontology are shown in Figure 2. The *Person* context

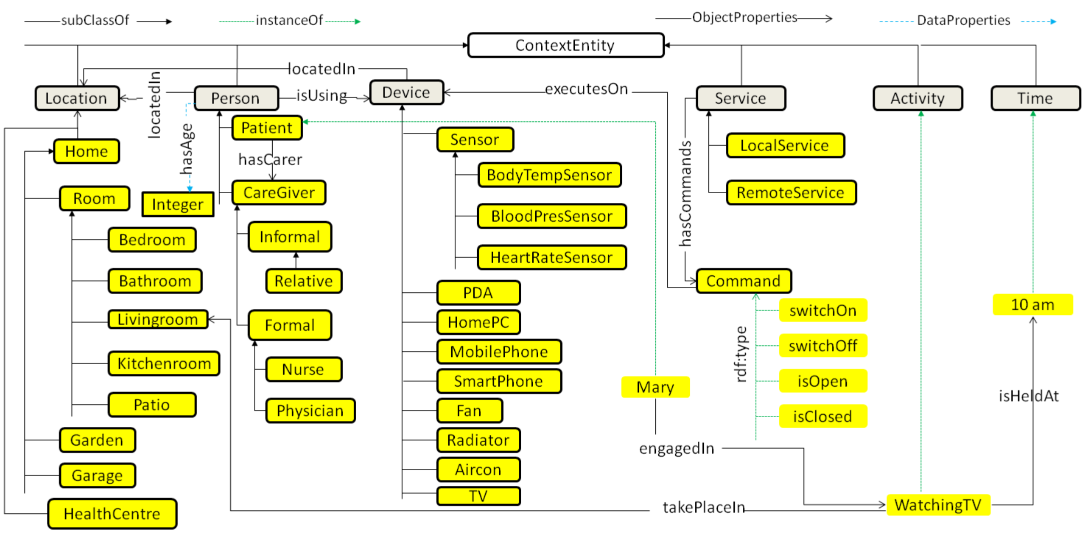


Fig. 1 A partial view of home environment and context ontology

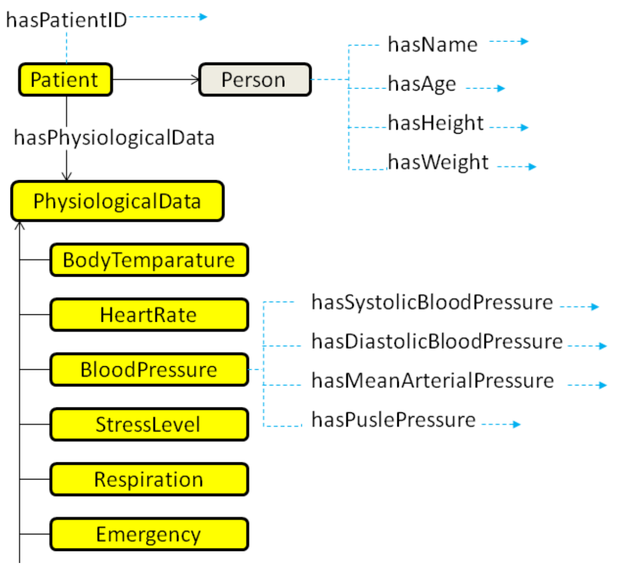
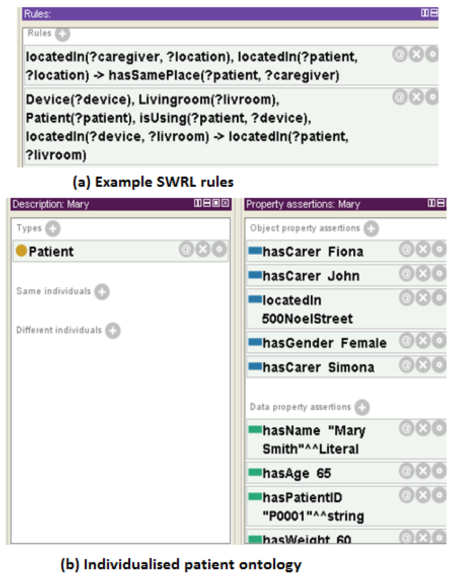


Fig. 2 A partial view of patient context ontology

(and hence *Patient* context) has person profile information such as, e.g., *Name, Age, Height*, and *Weight*, and the *Patient* context has *PhysiologicalData* context. For the design of context ontologies we use OWL 2 RL language and the Protégé ontology editor [20].

Note that there is no unique way to model a domain of discourse and there are always viable alternatives, however, a good way to think about modelling a domain depends on the application scenario one has in mind and its possible extensions that will be anticipated. Ontologies try to model real world scenario and therefore the concepts in the desired ontology must reflect this reality. It is an iterative process and many standard terms for health care domain already exist in the literature, for example, to model our domain a set of standard terms are obtained from SNOMED-CT [1], ICNP [15].


Rules:

Rules

```
locatedIn(?caregiver, ?location), locatedIn(?patient,
?location) -> hasSamePlace(?patient, ?caregiver)

Device(?device), Livingroom(?livroom),
Patient(?patient), isUsing(?patient, ?device),
locatedIn(?device, ?livroom) -> locatedIn(?patient,
?livroom)
```

(a) Example SWRL rules

Description: Mary

Types

● Patient

Same individuals

Different individuals

Property assertions: Mary

Object property assertions

hasCarer Fiona
hasCarer John
locatedIn 500NoelStreet
hasGender Female
hasCarer Simona

Data property assertions

hasName "Mary Smith"^^Literal
hasAge 65
hasPatientID "P0001"^^string
hasWeight 60

(b) Individualised patient ontology

Fig. 3 Example SWRL rules and individualised patient ontology

The combination of upper and domain ontologies described above, however only capture the static behaviour of a context-aware system. The context-aware systems modelled in our approach define their dynamic behaviour using Semantic Web Rule Language (SWRL). SWRL allows user to write rules using OWL concepts and its combination with OWL 2 RL provide more expressive language having greater deductive reasoning capabilities than OWL 2 RL alone. We can express more complex rule-based concepts using SWRL that cannot be modelled using OWL 2 RL. Some example rules are shown in Figure 3 (a). Thus our approach of ontological representation of context-aware systems gives a clean ontology design based on the distinction between the static information represented using OWL 2 RL and the dynamic aspects of the systems go into the SWRL rules.

2.2 Context-aware reasoning

We use ontology-based reasoning to determine concept satisfiability, subsumption relations, consistency and instance checking. That is, ontology-based reasoning infers implicit contexts from explicit contexts based on class relationships and property characteristics. For example, in our ontology, *hasCarer* is an inverse property of *takesCareOf*. From user defined explicit context *hasCarer(Mary, Fiona)* which states that *"Mary has a carer named Fiona"*, ontology-based reasoning can infer a new context *takesCareOf(Fiona, Mary)* which states that *"Fiona takes care of a patient named Mary"* based on the semantics *owl:inverseOf*. In Figure 3 (b), an instance of the patient domain ontology can be, for example, "Mary" has been shown. In this figure, we asserted some low-level contexts for a patient including *hasAge(Mary,65)*, *hasPatientID(Mary,P0001)* etc. It also includes some inferred contexts derived from context-reasoning using the DL reasoner Pellet.

As we have mentioned before, OWL DL is strictly limited to certain tree structure-like axioms and cannot be used to express arbitrary axioms, for example the relation between individuals with which an individual has relations cannot be expressed using OWL DL. The SWRL rules can remove such restrictions. Furthermore, low-level contexts can be transformed into meaningful information in terms of high-level contexts, where a set of suitable rules can exploit the real meaning of some raw values of context properties. Using SWRL rules we can have a flexible reasoning mechanism that will allow us inferring new contexts based on user defined rules. Thus, whenever certain changes are detected in its context, the system can be configured to change its behavior. Our aim is to build a context-aware system through distributed rule-based agents. Therefore, we translate OWL 2 RL ontology into a set of Horn clause rules, and the combination of these translated rules and the user defined SWRL rules (those are already in the Horn clause format) provide foundational knowledge to design the desired distributed rule-based agents. Moreover, user annotated Horn clause rules will enable reasoning with users' needs.

2.3 Translation of ontologies into rules

Since OWL 2 RL is based on DLP, the set of axioms and facts of an OWL 2 RL ontology can be translated to Horn clause rules [13]. In order to design an ontology-driven rule-based system, first we use the DLP framework [13] to translate an ontology to a set of Horn clause rules. In OWL 2 RL, facts are described using ClassAssertion and ObjectPropertyAssertion/DataPropertyAssertion which correspond to DL axioms of the form $a : C$ and $\langle a, b \rangle : P$, respectively, where a and b are individuals, C is a class, and P is an object/data property. Note that these facts are already in the Horn clause rule format with empty bodies.

The syntax of OWL 2 RL is asymmetric, i.e., the syntactic restrictions allowed for subclass expressions differ from those allowed for superclass expressions. For instance, an existential quantification to a class expression (*ObjectSomeValuesFrom*) is allowed only in subclass expressions whereas universal quantification to a class expression (*ObjectAllValuesFrom*) is allowed only in superclass expressions. These restrictions facilitate the translation of OWL 2 RL axioms into Horn clause rules based on the DLP framework. Translations of some of the OWL 2 RL axioms and facts into rules are given in Table 1. In the second column, complete DL statements are given which are constructed by the corresponding OWL 2 RL axioms and facts to illustrate the translation. For example, *ObjectIntersectionOf*($\sqcap$) is represented by the statement $C_1 \sqsubseteq D_1 \sqcap D_2$. The translation of SWRL rules is straightforward because they are already in the Horn clause rule format.

3 Context-aware systems as MASs

We model context-aware systems as multi-agent systems and rely on decentralised reasoning services and distribute context description to the agents (i.e., unlike many other agent-based context-aware systems we do not store context description managed by a central middleware). 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 $Ask(i, j, P)$ and $Tell(i, j, P)$ in their language, where i and j are agents and P is an atomic context not containing an Ask or a $Tell$. $Ask(i, j, P)$ means 'i asks j whether the context P is the case' and $Tell(i, j, P)$ means 'i tells j that

OWL 2 Axioms and Facts	DL Syntax	Horn clause rule
`ClassAssertions`	a:C	C(a)
`PropertyAssertion`	$\langle a,b\rangle : P$	$P(a,b)$
`SubClassOf`	$C \sqsubseteq D$	$C(x) \rightarrow D(x)$
`EquivalentClasses`	$C \equiv D$	$C(x) \rightarrow D(x)$,$D(x) \rightarrow C(x)$
`EquivalentProperties`	$P \equiv Q$	$Q(x,y) \rightarrow P(x,y)$
		$P(x,y) \rightarrow Q(x,y)$
`ObjectInverseOf`	$P \equiv Q^-$	$P(x,y) \rightarrow Q(y,x)$
		$Q(y,x) \rightarrow P(x,y)$
`TransitiveObjectProperty`	$P^+ \sqsubseteq P$	$P(x,y) \wedge P(y,z) \rightarrow P(x,z)$
`SymmetricObjectProperty`	$P \equiv P^-$	$P(x,y) \rightarrow P(y,x)$
`Object/DataUnionOf`	$C_1 \sqcup C_2 \sqsubseteq D$	$C_1(x) \rightarrow D(x)$, $C_2(x) \rightarrow D(x)$
`Object/DataIntersectionOf`	$C \sqsubseteq D_1 \sqcap D_2$	$C(x) \rightarrow D_1(x)$,$C(x) \rightarrow D_2(x)$
`Object/DataSomeValuesFrom`	$\exists P.C \sqsubseteq D$	$P(x,y) \wedge C(y) \rightarrow D(x)$
`Object/DataAllValuesFrom`	$C \sqsubseteq \forall P.D$	$C(x) \wedge P(x,y) \rightarrow D(y)$
`Object/DataPropertyDomain`	$\top \sqsubseteq \forall P^-.C$	$P(y,x) \rightarrow C(y)$
`Object/DataPropertyRange`	$\top \sqsubseteq \forall P.C$	$P(x,y) \rightarrow C(y)$

Table 1 Translation of OWL 2 RL axioms and facts into Horn clause rules

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., $P_1, P_2, \ldots, P_n \rightarrow Ask(i, j, P)$ whereas agent j may have an Ask or a $Tell$ with arguments (i, j, P) in the antecedent of the rule; e.g., $Tell(i, j, P) \rightarrow P$ 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 Asks 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 The 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 GCIs $C \sqsubseteq D$ 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 [4]. Our proposed approach is based on the work of [13] who show that a subset of DL languages can be effectively mapped into a set of Horn clause rules. Intuitively the set of translated rules corresponds to the $ABox$ joined with $TBox$ axioms (§2.3). 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 sTs' and $cp_i^{=n} \in V(s,i)$ and $cp_i^{=m} \in V(s',i)$ 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 $\alpha \in V(\pi_n, i)$,
- $\mathbb{M}, \pi, n \models cp_i^{=m}$ iff $cp_i^{=m} \in V(\pi_n, i)$,
- $\mathbb{M}, \pi, n \models \neg\varphi$ iff $\mathbb{M}, \pi, n \not\models \varphi$,
- $\mathbb{M}, \pi, n \models \varphi \wedge \psi$ 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 $\langle act_1, act_2, \ldots, act_{n_{A_g}} \rangle$ 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: $Rule_{i,r,\beta}$ (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 $\alpha \in V(s,j)$, $\alpha \notin V(s,i)$, $cp_i^{=m} \in V(s,i)$ for some $m < n_C(i)$, 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 a state $s \in S$ 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 $s' \in S$:

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 $Copy_{i,\alpha,\beta}, cp_i^{=m} \in V(s,i)$ for some $m \leq n_C(i)$: $V(s',i) = V(s,i) \setminus \{\beta, cp_i^{=m}\} \cup \{\alpha, cp_i^{=m+1}\}$;
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. $\forall s \in S$ and a tuple of actions $\langle act_1, act_2, \ldots, act_{n_{A_g}}\rangle$, 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, $\forall i \in A_g$;
3. $\forall s, s' \in S$, sTs' iff for some tuple of actions $\langle act_1, act_2, \ldots, act_{n_{A_g}}\rangle$, $act_i \in T_i(s)$ and the assignment in 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:

- $ByRule_i(P,m) = \neg B_iP \wedge cp_i^{=m} \wedge \bigvee_{r\in\Re\wedge cons(\bar{r}))=P}(B_ir \wedge \bigwedge_{Q\in ant(\bar{r})} B_iQ)$. This formula describes the state before the agent comes to believe formula P by the $Rule$ transition, m is the value of i's communication counter, P and Q are ground atomic formulas.
- $ByCopy_i(\alpha,m) = \neg B_i\alpha \wedge B_j\alpha \wedge cp_i^{=m-1}$, 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^* [24].

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 $\bigvee\limits_{n=0,\ldots,n_C(i)} cp_i^{=n}$, n is value of the communication counter of an agent i corresponding to its $Copy$ actions.

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

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

A6 $cp_i^{=m} \wedge \neg B_i\alpha \wedge B_j\alpha \rightarrow EX(B_i\alpha \wedge cp_i^{=m+1})$ where α is of the form $Ask(j,i,P)$ or $Tell(j,i,P)$, $i,j \in A_g$, $j \neq i$, $m < n_C(i)$. 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 $B_i\alpha \rightarrow AXB_i\alpha$ for any $\alpha \in S_M(i) \cup \Re$. This axiom states that an agent $i \in A_g$ always believes formulas residing in its static memory and its rules.

A9 $EX(B_i\alpha \wedge cp_i^{=m}) \rightarrow B_i\alpha \vee ByRule_i(\alpha,m) \vee ByCopy_i(\alpha,m)$ for any $\alpha \in \cup\Omega$. This axiom says that a new belief can only be added by one of the valid reasoning actions.

A10a $start \rightarrow cp_i^{=0}$ for all $i \in A_g$. At the start state, the agent has not performed any $Copy$ actions.

A10b $\neg EX\ start$. *start* holds only at the root of the tree.

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

A12 $\neg B_ir$ where $r \notin \Re$ and $i \in A_g$. This axiom tells agent i only believes its rules.

A13 $\varphi \rightarrow EX\varphi$, where φ does not contain $start$. This axiom describes an $Idle$ transition by all the agents.

A14 $\bigwedge_{i\in A_g} EX(\bigwedge_{\alpha\in\Gamma_i} B_i\alpha \wedge cp_i^{=m_i}) \rightarrow EX \bigwedge_{i\in A_g}(\bigwedge_{\alpha\in\Gamma_i} B_i\alpha \wedge cp_i^{=m_i})$ 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** - **A14**.

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 [24]. 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** and **A6**. 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 $\mathbb{M}, \pi, n \models B_i r \wedge \bigwedge_{P \in ant(\bar{r})} B_i P \wedge cp_i^{=m} \wedge \neg B_i cons(\bar{r})$, 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 $\mathbb{M}, \pi, n \models EX(B_i cons(\bar{r}) \wedge cp_i^{=m})$.

Let us consider **A6**. 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 $\mathbb{M}, \pi, n \models cp_i^{=m} \wedge \neg B_i \alpha \wedge B_j \alpha$, and $|V^M(s, i)| \leq n_M(i)$. Then $cp_i^{=m} \in V(\pi_n, i)$, $\alpha \notin V(\pi_n, i)$, and $\alpha \in V(\pi_n, j)$, for $i, j \in A_g$, $i \neq j$, and $m < n_C(i)$. This means that the action performed by i is $Copy_{i,\alpha,\beta}$. According to the definition of $\mathbb{M}(n_M, n_C)$, $\exists s' \in S \cdot \pi_n T s'$ and $V(s', i) = V(\pi_n, i) \setminus \{\beta, cp_i^{=m}\} \cup \{\alpha, cp_i^{=m+1}\}$. 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 $\mathbb{M}, \pi', n+1 \models B_i \alpha \wedge cp_i^{=m+1}$. Therefore, it is obvious that $\mathbb{M}, \pi, n \models EX(B_i \alpha \wedge cp_i^{=m+1})$.

Completeness can be shown by constructing a tree model for a consistent formula φ. This is constructed as in the completeness proof introduced in [24]. Then we use the axioms to show that this model is in $\mathbb{M}(n_M, n_C)$.

Since the initial state of all agents does not restrict the set of formulas they may derive in the future, for simplicity we conjunctively add to φ a tautology that contains all the potentially necessary formulas and message counters, in order to have enough sub-formulas for the construction. We construct a model $\mathbb{M} = (S, T, V)$ for

$$\varphi' = \varphi \wedge \bigwedge_{\alpha \in \Omega} (X B_i \alpha \vee \neg X B_i \alpha) \wedge \bigwedge_{n \in \{0 \ldots n_C(i)\}, i \in A_g} (X cp_i^{=n} \vee \neg X cp_i^{=n})$$

We then prove that $\mathbb{M}$ is in $\mathbb{M}(n_M, n_C)$ by showing that it satisfies all properties listed in Definition 4. Axioms **A3** and **A4** show that for any $i \in A_g$, there exists a unique $n \in \{0 \ldots n_C\}$ such that at a state s of $\mathbb{M}$, $cp_i^{=n} \in V(s, i)$.

At the root s_0 of (S, T), the construction of the model implies that there exists a maximally consistent set (MCS) Γ_0 such that $\Gamma_0 \supseteq V(s_0, i)$ and $start \in \Gamma_0$. Therefore, by axiom **A10**, it is trivial that $cp_i^{=0} \in V(s_0, i)$.

We then need to prove that $\forall s \in S$, $act_i \in T_i(s)$, and $i \in A_g$, $\exists s' \in S \cdot s T s'$ and $V(s', i)$ is the result of $V(s, i)$ after i has performed action act_i. Let us consider the case when act_i is $Rule_{i,r,\beta} \in T_i(s)$ for some $r \in \Re$. Since $Rule_{i,r,\beta}$ is applicable at s, $ant(\bar{r}) \subseteq V(s, i)$, $cons(\bar{r}) \notin V(s, i)$. Therefore there exists a MCS Γ such that $\Gamma \supseteq V(s, i)$, and $\bigwedge_{P \in ant(\bar{r})} B_i P \wedge cp_i^{=m} \wedge \neg B_i cons(\bar{r}) \in \Gamma$, for some $m \in \{0, \ldots, n_C\}$ and $|V^M(s, i)| \leq n_M(i)$. By axiom **A5** and Modus Ponens (MP), $EX(B_i cons(\bar{r}) \wedge cp_i^{=m}) \in \Gamma$. Therefore, according to the construction, $\exists s' \in S \cdot s T s'$, $V(s', i) \subseteq \Gamma'$ for some Γ', and $B_i cons(\bar{r}) \wedge cp_i^{=m} \in \Gamma'$. Therefore $V(s', i) = V(s, i) \setminus \{\beta\} \cup \{cons(\bar{r})\}$. For the $Copy_{i,\alpha,\beta} \in T_i(s)$ and $Idle_i \in T_i(s)$ actions, the proofs are similar by using MP and axioms **A6** and axiom **A13**. Then, using axiom **A14** we can show that, for any tuple of actions $\langle act_1, act_2, \ldots, act_{n_{A_g}} \rangle$, $act_i \in T_i(s)$ is applicable at $s \in S$ $\forall i \in A_g$, then $\exists s' \in S$ such that $V(s', i)$ is the result of $V(s, i)$ after performing act_i at s by agent i, $\forall i \in A_g$.

Finally, we prove that $\forall s, s' \in S \cdot s T s'$, $\exists$ a tuple of actions $\langle act_1, act_2, \ldots, act_{n_{A_g}} \rangle$ and $V(s', i)$ is the result of $V(s, i)$ when agent i performs act_i for all $i \in A_g$. By axioms **A7** and **A2**, $V(s', i)$ is different from $V(s, i)$ by at most one formula added and possibly a formula is removed. If no formula is added or removed, we consider act_i to be $Idle_i$. Let us now consider the case where a formula α is added. By axiom **A9**, if $cp_i^{=m} \in V(s, i)$ for some $m \in \{0, \ldots, n_C\}$ then either $cp_i^{=m}$ or $cp_i^{m+1} \in V(s', i)$. If $cp_i^{=m} \in V(s', i)$ then set act_i to be $Rule_{i,r,\beta}$ for some $r \in V(s, i)$, $\alpha = cons(\bar{r}) \notin V(s, i)$. If $cp_i^{=m+1} \in V(s', i)$ then set act_i to be $Copy_{i,\alpha,\beta}$. Thus, we proved the existence of the tuple $\langle act_1, act_2, \ldots, act_{n_{A_g}} \rangle$ for $s T s'$. Therefore, $\mathbb{M}$ is in $\mathbb{M}(n_M, n_C)$. □

5 Maude encoding

We build a multi-agent rule-based context-aware system whose rules are derived from the ontology of the smart space scenario described in Section 2, and the example system is adopted from [18,19]. The system consists of twelve agents, Figure 4 depicts smart space context-aware agents and their possible interactions. These are Blood Pressure Measurement Agent (1), Blood Sugar Measurement Agent (2), Heart Rate Measurement Agent (3), Smart Bed Agent (4), ACEInhibitor Box agent (5), Patient Monitor Agent (6), Communication Manager Agent (7), Palliative Care Unit (PCU) Coordinator Agent (8), General physician (formal care giver) Agent (9),

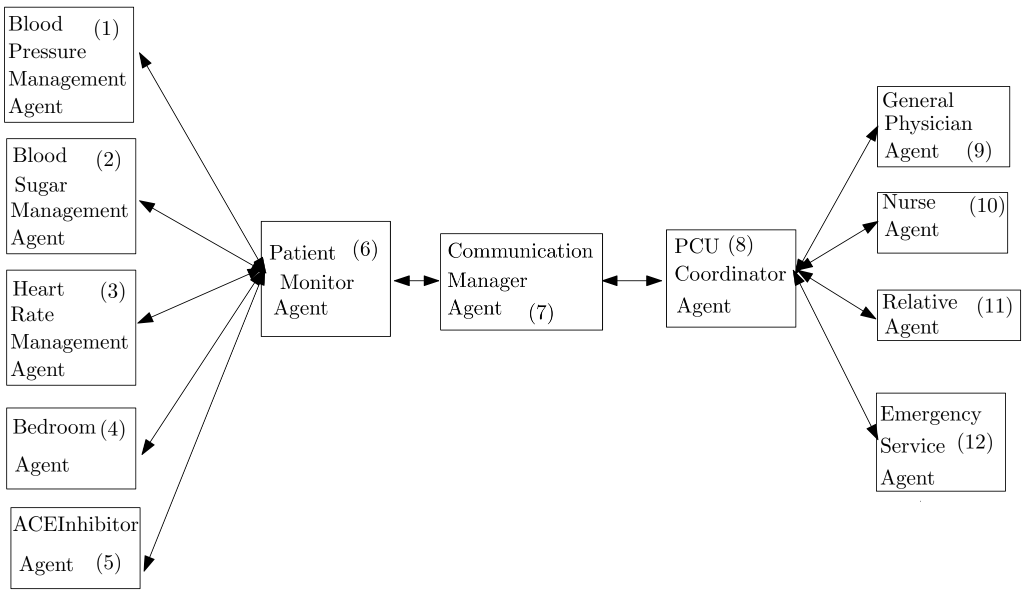

Blood Pressure Management Agent (1)
Blood Sugar Management Agent (2)
Heart Rate Management Agent (3)
Bedroom Agent (4)
ACEInhibitor Agent (5)
Patient Monitor Agent (6)
Communication Manager Agent (7)
PCU Coordinator Agent (8)
General Physician Agent (9)
Nurse Agent (10)
Relative Agent (11)
Emergency Service Agent (12)

Fig. 4 Agents and their possible interactions

General Nurse (formal care giver) agent (10), Relative (informal care giver) Agent (11) and Emergency Service Agent (12). Some of these agents, namely, $1, 2, 3, 4, 5$ and 6 are located at the smart home, $7, 8$ and 12 are located at the health centre, and the location of the care givers are not fixed. However, by adding more agents the system designer can make the system much more complex. 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. We omit the encoding here, however, it is similar to [23], apart from the implementation of agents memory bounds.

The measurement agents 1, 2, and 3 are able to infer high-level contexts from sensed low-level contexts using Horn clause rules in their KB. They can classify current blood pressure, blood sugar, and heart rate into different categories based on their current measurement values. E.g., agent 1's KB contains rules including the following:

Person(?p), hasSystolicBloodPressure(?p, ?sbp), hasDiastolicBloodPressure(?p, ?dbp), greaterThan(?sbp, '140), greaterThan(?dbp, '90) →hasBPCategory(?p, 'Hypertension);

Person(?p), hasSystolicBloodPressure(?p, ?sbp), hasDiastolicBloodPressure(?p, ?dbp), greaterThan(?sbp, '180), greaterThan(?dbp, '140) →hasBPCategory(?p, 'HypertensiveCrisis);

Person(?p), hasSystolicBloodPressure(?p, ?sbp), hasDiastolicBloodPressure(?p, ?dbp), greaterThan(?sbp, '90), greaterThan(?dbp, '60) →hasBPCategory(?p, 'Hypotension);

Person(?p), hasSystolicBloodPressure(?p, ?sbp), hasDiastolicBloodPressure(?p, ?dbp), greaterThan(?sbp, '90), greaterThan(?dbp, '60), lessThan(?sbp, '120), lessThan(?dbp, '90) →hasBPCategory(?p, 'Normal); and

hasBPCategory(?p, 'Hypertension)→Tell(1,6,hasBPCategory(?p, 'Hypertension)).

The first rule classifies that the person has blood pressure category *Hypertension* if her Systolic Blood Pressure is greater than 140 and Diastolic Blood Pressure is greater than 90. That is, agent 1 may infer high-level context *hasBPCategory('Mary, 'Hypertension)* from the low-level contexts, e.g., *hasSystolicBloodPressure('Mary, '145)*, *hasDiastolicBloodPressure('Mary, '95)*, and so on. The fifth rule is a communication rule of agent 1 through which it interacts with agent 6 and passes the context *hasBPCategory('Mary, 'Hypertension)* when it believes that *Mary* has *Hypertension* at the moment. Similar to the above, agent 1 has other three communication rules for other categories.

In a similar fashion, agent 2 may infer context *hasDBCategory('Mary, 'EstablishedDiabetes)* if Blood Sugar Level Before Meal is greater than 126, and agent 3 may infer *hasHRCategory('Mary, 'BelowAverage)* if current Heart Rate is greater than 74ft and less than 80, and so on. Agent 2 and 3's KBs contain rules including the following:

Person(?p), hasBloodSugarLevelBeforeMeal(?p, ?bsl), greaterThan(?bsl, '126) → hasDBCategory(?p, 'EstablishedDiabetes); and

hasDBCategory(?p, 'EstablishedDiabetes) → Tell(2,6,hasDBCategory(?p, 'EstablishedDiabetes)).

Person(?p), hasHeartRate(?p, ?hrt), greaterThan(?hrt, '74), lessThan(?hrt, '80) →hasHRCategory(?p, 'BelowAverage); and *hasHRCategory(?p,'BelowAverage)→ Tell(3,6,hasHRCategory(?p, 'BelowAverage)).*

Once these agents infer high level contexts they can interact with the patient monitor agent 6 and pass those information. Agent 6's KB contains rules including:
Person(?p), hasPatientID(?p,?pid), PatientID(?pid)→ Patient(?p);
Tell(1,6,hasBPCategory(?p,'Hypertension)) → hasBPCategory(?p, 'Hypertension);
Patient(?p), hasBPCategory(?p,'HypertensiveCrisis), hasDBCategory(?p, 'EstablishedDiabetes), hasHRCategory(?p,'Poor) → hasAlarmLevel(?p, 'VeryHigh);
Patient(?p), hasBPCategory(?p,'HypertensiveCrisis), hasDBCategory(?p, 'EstablishedDiabetes), hasHRCategory(?p,'BelowAverage) → hasAlarmLevel(?p,'High);
hasAlarmLevel(?p,'High)) → hasPrescribedDrug(?p,'ACEInhibitor);
hasPrescribedDrug(?p,'ACEInhibitor) → Tell(6,5, ACEInhibitorBox('SwitchedON)); and
hasAlarmLevel(?p,'High)→Tell(6,7,hasAlarmLevel(?p,'High)).

The first rule checks whether the person is a patient. The second rule is a trust rule for agent 6 that causes it to believe agent 1 when agent 1 informs it that context, for example, *hasBPCategory('Mary, 'Hypertension)*. Agent 6 may infer various other contexts depending on the current contexts it has received from those measurement agents. The patient monitor agent also classifies the alarm levels based on the contexts and interacts with other agents. Using the second rule, agent 6 infers the context, for example, *hasPrescribedDrug('Mary, 'ACEInhibitor)* which prescribes the drug *ACEInhibitor* to *Mary*. Using the third rule agent 6 interacts with agent 5 and informs that the indicator of the ACEInhinitor Box must be switched on, so that patient can know she has to take ACEInhibitor at this moment. The patient monitor agent also interacts with the communication manager agent 7 which is located in the health centre. Once the communication manager agent receives the alarming information from agent 6, it communicates with the PCU coordinator agent. Depending on the alarm level, PCU coordinator agent interacts with various agents including general physician, nurse, relative and emergency service. For example, if the alarm level is *VeryHigh*, PCU coordinator directly informs to the emergency service. If the alarm level is *Low*, PCU coordinator only informs to the relative. If the alarm level is *High*, PCU coordinator interacts with the nurse and waits for a response. If it receives *NotAvailalabe* or *Busy* response, then it interacts with the general physician, and waits for a response, if it also receives *NotAvailalabe* or *Busy* response from the physician, then PCU coordinator informs the patient's alarming level to the emergency service, and so on. For the sake of brevity, we are unable to represent example rules of other agents in the system. In order to model this scenario we have derived 176 Horn clause rules from the smart space ontology and distributed them to the agents as working memory facts and knowledge base rules. E.g., the KB of the blood pressure measurement agent contains 8 rules, heart rate measurement agent is modelled using 12 rules, patient monitor agent is modelled using 30 rules, PCU coordinator agent is modelled using 36 rules and so on. We verified a number of interesting resource-bounded properties of the system including the following:

$$G(B_6\, Tell(1, 6, hasBPCategory('Mary,' Hypertension)) \wedge B_6\, Tell(2, 6, hasDBCategory(?p,' EstablishedDiabetes)) \wedge B_6\, Tell(3, 6, hasHRCategory(?p,' BelowAverage)) \rightarrow X^n B_6\, Tell(6, 7, hasAlarmLevel('Mary,' High))$$

the above property specifies that whenever agents 1, 2, and 3 tell agent 6 that the blood pressure, diabetes, and heart rate categories are Hypertension, EstablishedDiabetes , and BelowAverage, respectively, of the patient Mary, within n time steps agent 6 sending a classified message to agent 7 that the alarm level of the patient Mary is High.

$$G(B_8\, hasAlarmLevel('Mary,' High) \wedge B_8\, Tell(9, 8, hasCareStatus('John,' NotAvailable)) \rightarrow X^n \;\; B_8\, Tell(8, 12, hasAlarmLevel('Mary,' High)))$$

the above property specifies that whenever alarm level of the patient Mary is High and the PCU coordinator agent has interacted the physician Dr John and received acknowledgement as NotAvailable, then the PCU coordinator agent informs the patient's alarm level to the emergency service within n timesteps.

$$G(B_6\, hasAlarmLevel('Mary,' High) \rightarrow X^n \;\; B_6\, Tell(6, 5, ACEInhibitorBox('SwitchedON)) \wedge cp_6{}^{=m})$$

which specifies that whenever agent 6 classifies that alarm level of the patient Mary is High, within n time steps it tells agent 5 to switch on the indicator of the ACEInhibitor Box, so that Mary should know she has to take ACEInhibitor, while exchanging m messages ($cp_6{}^{=m}$ states that the value of agent 6's communication counter is m).

The above properties are verified as true when the value of n is 7 in the first property, value of n is 3 in the second property, and the values of n and m are 4 and 5 in the third property. However, the properties are verified as false and the model checker returns counterexamples when we assign a value to n which is less than 7 in the first property, less than 3 in the second property, and values to n and m which are less than 4 and 5 in the third property. While verifying the first property, agents 1, 2, and 3 have the same dynamic memory size of 2 units and the memory size of the agent 6 is 6 units. For the second property, agent 8 requires 3 units and for the third property agent 6 requires 4 units of memory space for their reasoning tasks. These are the minimal space requirements for their reasoning while verifying above properties. Note that, while verifying a particular property, the amount of space needed for the reasoning task for an agent(s) may vary for other properties. Therefore, a system designer has to decide what are the desired properties of the system that should be verified, and what would be an optimal use of resources for that problem.

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 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. 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 agents considering non-monotonic reasoning. The logic developed in this paper is based on monotonic reasoning where beliefs of an agent cannot be revised based on some contradictory evidence. We would like to extend this logic considering defeasible reasoning which is a simple rule-based technique used to reason with incomplete and inconsistent information [6].

References

1. SNOMED-CT Systematized Nomenclature of Medicine-Clinical Terms. http://www.ihtsdo.org/snomed-ct/ (2007)
2. Agarwal, S., Joshi, A., Finin, T., Yesha, Y., Ganous, T.: A pervasive computing system for the operating room of the future. Journal of Mobile Networks and Applications **12**(2-3), 215–228 (2007)
3. Alechina, N., Jago, M., Logan, B.: Modal logics for communicating rule-based agents. In: Proceedings of the 17th European Conference on Artificial Intelligence A, pp. 322–326 (2006)
4. 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.) MoChArt2008, *LNCS*, vol. 5348, pp. 1–14. Springer,Heidelberg (2008)
5. 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)
6. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Transactions on Computational Logic **2**(2), 255–287 (2001)
7. Bardram, J.E., , Nørskov, N.: A context-aware patient safety system for the operating room. In: Proceedings of of the 10th international conference on Ubiquitous computing, pp. 272–281 (2008)
8. Bardram, J.E.: Hospitals of the future ubiquitous computing support for medical work in hospitals. In: Proceedings of UbiHealth'03 the 2nd international workshop on ubiquitous computing for pervasive healthcare applications (2003)
9. Bechhofer, S., van Harmelen, F., Hendler, J.A., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference, World Wide Web Consortium, recommendation rec-owl-ref-20040210 (2004)
10. Dey, A., Abwowd, G.: Towards a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22, Georgia Institute of Technology
11. Eker, S., Meseguer, J., Sridharanarayanan, A.: The maude LTL model checker and its implementation. In: Ball, T., Rajamani, S.K. (eds.) SPIN2003, *LNCS*, vol. 2648, pp. 230–234. Springer-Verlag (2003)
12. Esposito, A., Tarricone, L., Zappatore, M., Catarinucci, L., Colella, R.: A framework for context-aware home-health monitoring. International Journal of Autonomous and Adaptive Communications Systems **3**(1), 75–91 (2010)
13. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: Combining logic programs with description logic. In: WWW2003, pp. 48–57. ACM Press (2003)
14. Gruber, T.: A translation approach to protable ontology specifications. Knowledge Acquisition - Special issue: Current issues in knowledge modeling **5**(2), 199–220 (1993)
15. Hardiker, N., Coenen, A.: A formal foundation for ICNP. Journal of Stud Health Technol Inform **122**, 705–709 (2006)
16. 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 (2004)
17. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. Web Semantics: Science, Services and Agents on the World Wide Web **3**(2-3), 79–115 (2005)
18. Moreno, A.: Medical applications of multi-agent systems. Computer Science & Mathematics Department, Universitat Rovira i Virgili ETSE. (2007)
19. Paganelli, F., Giuli, D.: An ontology-based context model for home health monitoring and alerting in chronic patient care networks. In: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops, 2007, pp. 838–845 (2007)
20. Protégé: The Protégé ontology editor and knowledge-base framework (Version 4.1). http://protege.stanford.edu/ (2011)
21. Rakib, A.: Verifying requirements for resource-bounded agents. Ph.D. thesis, The University of Nottingham. (2011)
22. 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)
23. Rakib, A., Faruqui, R.U.: A Formal Approach to Modelling and Verifying Resource-Bounded Context-Aware Agents. In: P.C. Vinh et al. (Eds.) ICCASA'12, *LNICST*, vol. 109, pp. 86–96. Springer-Verlag (2013)
24. Reynolds, M.: An axiomatization of full computation tree logic. J. Symb. Log. **66**(3), 1011–1057 (2001)
25. Uschold, M., Gruninger, M.: Ontologies: Principles, Methods and Applications. Knowledge Engineering Review **11**(2) (1996)
26. Viterbo F, J., da G. 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)
27. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using owl. In: PerCom Workshops2004, pp. 18–22 (2004)
28. Weiser, M.: The computer for the 21st century. ACM SIGMOBILE Mobile Computing and Communications Review– Special issue dedicated to Mark Weiser **3**(3), 3–11 (1999)
29. Wooldridge, M.: An Introduction to Multi-Agent Systems. John Wiley & Sons Inc (2009)