A Logic for Context-Aware Non-monotonic Reasoning Agents

Abdur Rakib and Hafiz Mahfooz Ul Haque

School of Computer Science
The University of Nottingham, Malaysia Campus
{Abdur.Rakib,khyx2hma}@nottingham.edu.my

Abstract. We develop a logical model for resource-bounded context-aware multi-agent systems which handles inconsistent context information using non-monotonic reasoning. We extend the temporal logic CTL^* with belief and communication modalities, and the resulting logic $\mathcal{L}_{DROCS}$ allows us to describe a set of rule-based non-monotonic context-aware agents with bounds on computational (time and space) and communication resources. We use OWL 2 RL ontologies and Semantic Web Rule Language (SWRL) for context-modelling and rules that enables the construction of a formal system. We provide an axiomatization of the logic and prove it is sound and complete. We illustrate the use of the logical model on a simple example.

Keywords: Context-aware, Rule-based reasoning, Defeasible reasoning, Multi-agent systems, Ontology.

1 Introduction

The term context-awareness in pervasive computing has been used relatively long ago, e.g., by Schilit and colleagues [24]. It describes the ability of a device to realise a situation and act accordingly. Thus a system is said to be context-aware if it can extract, interpret, and is able to adapt its behaviour to the current context of use [6]. In the literature, several definitions of *context* have been proposed including those presented by [25,8]. 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 [8]. A context can be formally defined as a $\langle subject, predicate, object\rangle$ 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 [8], *"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 fever categorized as High"* as $\langle Mary, hasFever, High\rangle$ and *"Mary has a carer named Fiona"* as $\langle Mary, hasCarer, Fiona\rangle$. Here, the caregiver of a patient is dynamically identified based on the care status of the caregiver. These contexts can be written using

A. Gelbukh et al. (Eds.): MICAI 2014, Part I, LNAI 8856, pp. 453–471, 2014.

first order predicates as *hasFever('Mary, 'High)* and *hasCarer('Mary, 'Fiona)* respectively.

In the literature, various techniques have been proposed to develop context-aware systems, including rule-based techniques [7,12]. In rule-based techniques 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. An agent is called a rule-based agent if its behaviour and/or its knowledge is expressed by means of rules. In the setting of this paper, we call a *rule-based agent* as *context-aware agent* since it enables the system to understand and process context information expressed using first order rules. While active context-aware computing leads to a new paradigm that leverages interaction among system users and their environments, many challenges might arise on the basis of computational (time:measured in number of computational steps and space: amount of memory) and communication (number of messages need to be exchanged between devices) resources. This is due to the fact that many context-aware systems often run on tiny resource-bounded devices and in highly dynamic environments. The state of the art context-aware capable devices including PDAs, mobile phones, smart phones, GPS system, and wireless sensor nodes usually operate under strict resource constraints, e.g., battery energy level, memory, processor, and quality of wireless connection [26]. Therefore, for a given set of context-aware reasoning agents with some inferential abilities and computational (time and space) and communication resource bounds, it may not be clear whether a desired context can be inferred and if it can what computational and communication resources must be devoted by each agent. Furthermore, reasoning tasks may involve complex processing and resolve conflicting context information. Although research advances have been made in building context-aware systems for many applications, however well developed theoretical foundations considering their resource-boundedness features are still lacking. In this work, we present a logical framework for resource-bounded context-aware non-monotonic reasoning agents which is intended to be theoretically well motivated and technically well defined. We develop a logic $\mathcal{L}_{DROCS}$ which extends the temporal logic CTL^* with belief and communication modalities and incorporates defeasible reasoning [19] technique (one of most prominent member of the non-monotonic reasoning techniques). The logic $\mathcal{L}_{DROCS}$ allows us to describe a set of context-aware non-monotonic reasoning agents with bounds on computational and communication resources. We provide an axiomatization of the logic and prove it is sound and complete, and using a simple example we show how we can express some interesting resource-bounded properties of a desired system.

The rest of the paper is organised as follows. In section 2, we briefly review description logics, ontologies and defeasible reasoning. In section 3, we describe our model of context-aware systems. In section 4, we develop the logic $\mathcal{L}_{\mathcal{DROCS}}$ and show an illustrative example system, in section 5 we present related work, and conclude in section 6.

2 Preliminaries

2.1 Description Logics and Ontology

Description logics (DLs) are a well-known family of knowledge representation formalisms that can be used to represent knowledge of a domain in a structured and organized manner [3]. A DL is based on the notion of concepts (classes) and roles (binary relations), and is mainly characterized by the constructors that let complex concepts and roles to be constructed from atomic ones. A DL knowledge base ($\mathcal{KB}$) has two components: the Terminology Box ($TBox$) and the Assertion Box ($ABox$). 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 $a : C$ (or $C(a)$ which states that the individual a is an instance of the class C) and role assertions in the form of $\langle a, b\rangle : R$ (or $R(a, b)$ which states that the individual a is related to the individual b through the relation R). The ability to model a domain and the decidable computational characteristics make DLs the basis for the widely accepted ontology languages such as Web Ontology Language (OWL). The Web Ontology Language version 2, OWL 2, has three sub-languages: OWL 2 EL, OWL 2 QL, and OWL 2 RL [17]. OWL 2 RL is suitable for rule-based applications, it enables additional rules such as SWRL to be added to ontologies for more expressive descriptions of an application domain.

There have been various approaches proposed for context modelling, however, ontology-based approach has been advocated as being the most promising one [4]. For example, if we are interested in modelling a smart space health care monitoring system we may use OWL concept names to capture terms that are relevant to this domain and ultimately represent the desired scenario using a set of logical statements [12]. We model context-aware systems using OWL 2 RL ontologies (and SWRL) and extract rules from an ontology following a similar approach proposed by [13] to design our rule-based non-monotonic context-aware agents. We developed a translator that takes as input an OWL 2 RL ontology in the OWL/XML format (an output file of the Protégé [20] editor) and translates it to a set of plain text rules. We use the OWL API [15] to parse the ontology and extract the set of axioms and facts. The design of the OWL API is directly based on the OWL 2 Structural Specification and it treats an ontology as a set of axioms and facts which are read using the visitor design pattern. We also extract the set of SWRL rules using the OWL API which are already in the Horn clause rule format. First, atoms with corresponding arguments associated with the head and the body of a rule are identified and we then generate a plain text Horn clause rule for each SWRL rule using these atoms. Abox axioms are already in Horn clause formats as well and they are simply rules with empty bodies.

2.2 Defeasible Reasoning

Defeasible reasoning is a simple rule-based reasoning technique that has been used to reason with incomplete and inconsistent information [2]. A defeasible logic theory consists of a collection of rules that reason over a set of facts to reach a set of defeasible conclusions. It also supports priorities among rules to resolve conflicts. More formally, a defeasible theory $\mathcal{D}$ is a triple $(\Re, \mathcal{F}, \succ)$ where $\Re$ is a finite set of rules, $\mathcal{F}$ is a finite set of facts, and $\succ$ is a superiority relation on $\Re$. The superiority relation $\succ$ is often defined on rules with complementary heads and its transitive closure is irreflexive, i.e., the relation $\succ$ is acyclic. Rules are defined over literals, where a literal is either a first-order atomic formula P or its negation $\neg P$. For example, given a literal l, the complement $\sim l$ of l is defined to be P if l is of the form $\neg P$, and $\neg P$ if l is of the form P. In the rules, we assume variables are preceded by a question mark and constants are preceded by a single quote. In $\mathcal{D}$, there are three kinds of rules those are often represented using three different arrows.

Strict rules are of the form: $P_1, P_2, \ldots, P_n \rightarrow P$ where the conclusion P is valid whenever its antecedents $P_1, P_2, \ldots P_n$ are true. An example of a strict rule can be *"A person who has a patient identification number is a patient"* which can be written as **r1:** *Person(?p), PatientID(?pid), hasPatientID(?p, ?pid)* $\rightarrow$ *Patient(?p).*

Defeasible rules are of the form: $P_1, P_2, \ldots, P_n \Rightarrow P$ and they can be defeated by contrary evidence. An example rule can be **r2:** *Patient(?p), hasFever(?p, 'High)* $\Rightarrow$ *hasSituation(?p, 'Emergency).* This rule states that if the patient has a high fever then there are provable reasons to declare an emergency situation for her, unless there is other evidence that provides reasons to believe the contrary. For example, a defeasible rule **r3:** *Patient(?p), hasFever(?p, 'High), hasConsciousness(?p, 'Yes)* $\Rightarrow \sim$ *hasSituation(?p, 'Emergency).* We can observe that the defeasible rule **r3** is more specific (we assume that **r3** is superior to **r2** i.e., $\mathbf{r3} \succ \mathbf{r2}$) and it could override the rule **r2**. That is a defeasible rule is used to represent tentative information that may be used if nothing could be placed against it.

Defeater rules are of the form: $P_1, P_2, \ldots, P_n \rightsquigarrow P$ and they don't support inferences directly, however, they can be used to block the derivation of inconsistent conclusions. Their only use is to prevent conclusions. For example, **r4:** *Patient(?p), hasFever(?p, 'High), hasDBCategory(?p, 'EstablishedDiabetes* $\rightsquigarrow$ *hasSituation(?p, 'Emergency).*

A rule can have multiple (ground) instances. For example, *Person('Mary), PatientID('P001), hasPatientID('Mary, 'P001)* $\rightarrow$ *Patient('Mary)* could be one possible instance of the rule **r1**. In the above rules, suppose the superiority relation $\succ$ among the rules are defined as follows $\mathbf{r1} \succ \mathbf{r4}$, $\mathbf{r4} \succ \mathbf{r3}$, $\mathbf{r3} \succ \mathbf{r2}$ and the current set of facts (contexts) are *Person('Mary), PatientID('P001), hasPatientID('Mary, 'P001), hasFever('Mary, 'High), hasConsciousness('Mary, 'Yes)* then by matching and firing those rules a defeasible conclusion $\sim$ *hasSituation('Mary, 'Emergency)* can be inferred.

3 Context-Aware Systems as Multi-agent Defeasible Reasoning Systems

We model a context-aware system as a multi-agent defeasible reasoning system which consists of $n_{Ag}(\geq 1)$ individual agents $A_g = \{1, 2, ..., n_{Ag}\}$. Each agent $i \in A_g$ is represented by a triple $(\Re, \mathcal{F}, \succ)$, where $\mathcal{F}$ is a finite set of facts contained in the working memory, $\Re = (\Re^s, \Re^d)$ is a finite set of strict and defeasible rules representing the knowledge base, and $\succ$ is a superiority relation on $\Re$. As we have mentioned in the preceding section, rules are of the form $P_1, P_2, \ldots, P_n \hookrightarrow P$ (derived from OWL 2 RL and SWRL with possible user annotation), and a working memory contains ground atomic facts (contexts) taken from ABox representing the initial state of the system. Without loss of generality, in the rest of this paper we assume $\hookrightarrow$ as either $\rightarrow$ or $\Rightarrow$. In a rule instance, the antecedents $P_1, P_2, \ldots, P_n$ and the consequent P are context information. The antecedents of a rule instance form a complex context which is a conjunction of n contexts. We say that two contexts are contradictory iff they are complementary with respect to $\sim$, for example, *hasSituation('Mary, 'Emergency)* and $\sim$*hasSituation('Mary,'Emergency)* are contradictory contexts. Note that in our model the set of facts translated from the ABox needs to be consistent, i.e., if it contains pair of contradictory contexts then they can be detected and removed. We assume that the set $\Re^s$ of strict rules is non-contradictory which is used to represent non-defeasible contextual information, however, the set $\Re^d$ of defeasible rules is contradictory and hence the set $\Re$ which is $\Re^s \cup \Re^d$ may also be contradictory. Conflicting contexts may be resolved using the superiority relation $\succ$ among rules. An agent i can fire instances of strict rules to infer new non-contradictory contexts, while a defeasible context P can be inferred if there is a rule instance whose consequence is P and there does not exist a stronger rule instance whose consequence is $\sim P$. Since the translated rules from an ontology are not prioritized, we assume that the rule priorities are provided by the system designers, depending on the intended applications. We further assume that the priorities are static, i.e., the rule firing constraint does not change during the reasoning process. 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 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 *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 [20] editor all the arguments of *Ask* and *Tell* are represented using constant symbols.

```
Rule ::= Atoms '↪' Atom | ∼ Atom
Atoms ::= Atom {, Atom}*
Atom ::= standardAtom | commmunicationAtom
standardAtom ::= description'('i-object ')'
            | individualvaluedProperty'('i-object ',' i-object ')'
            | datavaluedProperty'('i-object ',' d-object ')'
            | sameIndividuals'('i-object ',' i-object ')'
            | differentIndividuals'('i-object ',' i-object ')'
            | dataRange'(' d-object ')'
            | builtIn'(' builtinId ',' {d-object}* ')'
communicationAtom ::= 'Ask(' i ',' j ',' standardAtom ')'
            | 'Tell(' i ',' j ',' standardAtom ')'
i ::= 1 | 2 | ... | nAg
j ::= 1 | 2 | ... | nAg
builtinID ::= URIreference
i-object ::= i-variable | individualID
d-object ::= d-variable | dataLiteral
i-variable ::= 'I-variable('URIreference')'
d-variable ::= 'D-variable('URIreference')'
```

Listing 1.1. Abstract syntax of rules

4 The Logic $\mathcal{L}_{DROCS}$

We now introduce the logic $\mathcal{L}_{DROCS}$ based on [1] (which has been developed on propositional language and monotonic reasoning). 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 strict and defeasible rules. Intuitively, the set of translated rules corresponds to the *ABox* joined with *TBox* axioms of an OWL 2 RL ontology and SWRL rules. Let us define the internal language of each agent in the system. Let the set of agents be $A_g = \{1, 2, ..., n_{Ag}\}$, $\mathcal{C} = \{C_1, C_2, \ldots C_l\}$ be a finite set of concepts, $\mathcal{R} = \{R_1, R_2, \ldots, R_m\}$ be a finite set of roles. We also define a set $\mathcal{Q} = \{Ask(i, j, P), Tell(i, j, P)\}$, where $i, j \in A_g$ and $P \in \mathcal{C} \cup \mathcal{R}$. Let $\Re^s$ be a finite set of strict rules and $\Re^d$ be a finite set of defeasible rules. Let $\Re = \Re^s \cup \Re^d = \{r_1, r_2, \ldots, r_n\}$ be a finite set of rules of the form $P_1, P_2, \ldots, P_t \hookrightarrow P$, where $t \geq 0$, $P_i, P \in \mathcal{C} \cup \mathcal{R} \cup \mathcal{Q}$ for all $i \in \{1, 2, \ldots, t\}$,$P_i \neq P_j$ for all $i \neq j$, and $\hookrightarrow$ as either $\rightarrow$ or $\Rightarrow$. More specifically, P_i and P are OWL atoms of the following form: $C_i(x)$ and $R_j(y, z)$. Where $C_i \in \mathcal{C}$, and x is either a variable, an individual or a data value. $R_j \in \mathcal{R}$, when it is an Object property y, z are either variables, individuals or data values, however, y is variable or individual and z is a data value when R_j is a Datatype property. In Listing 1.1, we specify the abstract syntax of rules using a BNF. In this notation, the terminals are quoted, the non-terminals are not quoted, alternatives are separated by vertical bars, and components that can occur zero or more times are enclosed braces followed by a superscript asterisk

symbol ($\{\ldots\}^*$). A class atom represented by `description(i-object)` in the BNF consists of an OWL 2-named class and a single argument representing an OWL 2 individual, for example an atom `Person(a)` holds if `a` is an instance of the class description `Person`. Similarly, an individual property atom represented by `individualvaluedProperty(i-object,i-object)` consists of an OWL 2 object property and two arguments representing OWL 2 individuals, for example an atom `hasCarer(a,b)` holds if `a` is related to `b` by property `hasCarer` and so on.

For convenience, we use the notation $ant(r)$ for the set of antecedents of r and $cons(r)$ for the consequent of r, where $r \in \Re$. We fix a finite set of variables X and a finite set of constants D and assume δ is some substitution function from the set of variables of a rule into D. We denote by $\mathcal{G}(\Re)$ the set of all the ground instances of the rules occurring in $\Re$, which is obtained using δ (a more formal definition is given in Definition 2). Thus $\mathcal{G}(\Re)$ is finite. Let $\bar{r} \in \mathcal{G}(\Re)$ be one of the possible instances of a rule $r \in \Re$. $C(a)$, $R(a,b)$, $Ask(i,j,C(a))$, $Ask(i,j,R(a,b))$, $Tell(i,j,C(a))$, and $Tell(i,j,R(a,b))$ are ground atoms, for all $C \in \mathcal{C}, R \in \mathcal{R}$. The internal language $\mathcal{L}$ includes all the ground atoms and rules. Let us denote the set of all formulas (rules and ground atoms) by Ω which is finite. In the language of $\mathcal{L}$ we have a 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 ground atom. The static part contains initial information to start up the systems, e.g., initial working memory facts, thus it's size is determined by the number of initial facts. The dynamic part contains newly derived facts as the system moves. Only facts stored in $D_M(i)$ may get overwritten, and this happens if an agent's memory is full or a contradictory context arrives in the memory (even if the memory is not full). Whenever newly derived context arrives in the memory, it is compared with the existing contexts to see if any conflict arises. If so then the corresponding contradictory context will be replaced with the newly derived context, otherwise an arbitrary context will be removed if the memory is full. Note that unless otherwise stated, in the rest of the paper we shall assume that memory means $D_M(i)$.

The syntax of $\mathcal{L}_{\mathcal{DROCS}}$ 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 (wffs) of $\mathcal{L}_{\mathcal{DROCS}}$;
- $cp_i^{=n}$ (which states that the value of agent i's communication counter is n) is a wff of $\mathcal{L}_{\mathcal{DROCS}}$ 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 $\mathcal{L}_{\mathcal{DROCS}}$ 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 $\mathcal{L}_{\mathcal{DROCS}}$ 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 $\mathcal{L}_{\mathcal{DROCS}}$, then so are $\neg\varphi$ and $\varphi \wedge \psi$;
- If φ and ψ are wffs of $\mathcal{L}_{\mathcal{DROCS}}$, 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.

For convenience, we define the following sets: $CP_i = \{cp_i^{=n} \mid n = \{0,\ldots,n_C(i)\}\}$, $CP = \bigcup_{i \in A_g} CP_i$. Now we define priority relation between rules as follows.

Definition 1 (Rule priority). *Let $pri : \Re \rightarrow N_{\geq 0}$ be a function that assigns each rule a non-negative integer. We define a partial order $\succ$ on $\Re$ such that for any two rules $r, r' \in \Re$ we say that $r \succ r'$ (rule r has priority over r') iff $pri(r) \geq pri(r')$, where $\geq$ is the standard greater-than-or-equal relation on the set of non-negative integers $N_{\geq 0}$.*

The semantics of $\mathcal{L}_{\mathcal{DROCS}}$ is defined by $\mathcal{L}_{\mathcal{DROCS}}$ 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., $\forall s \in S \cdot \exists s' \in S \cdot sTs'$. (S,T) is a ω-tree frame iff the following conditions are satisfied.

1. S is a non-empty set and T is total;
2. Let $<$ be the strict transitive closure of T, namely $\{(s,s') \in S \times S \mid \exists n \geq 0, s_0 = s, \ldots, s_n = s' \in S$ such that $s_iTs_{i+1} \forall i = 0,\ldots,n-1\}$;
3. For all $s' \in S$, the past $\{s \in S \mid s < s'\}$ is linearly ordered by $<$;
4. There is a smallest element called the root, which is denoted by s_0;
5. Each maximal linearly $<$- ordered subset of S is order-isomorphic to the natural numbers.

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 $\mathcal{L}_{\mathcal{DROCS}}$ 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 $V^M(s,i) = \{\alpha | \alpha \in V^B(s,i) \cap D_M(i)\}$ which represents the set of facts stored in the dynamic memory of agent i at state s. 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 $r : P_1, P_2, \ldots, P_n \hookrightarrow P$ 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 $i \in A_g, V(s',i) = V(s,i) \setminus \{\beta\} \cup \{cons(\bar{r})\} \cup \{Ask(j,i,C(a))\} \cup \{Tell(j,\ i, C(a)\} \cup \{Ask(j,i,R(a,b))\} \cup \{Tell(j,i,R(a,b)\}$. 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. Where β may be an arbitrary context or a contradictory context which can be replaced depending on the status of the memory and the newly derived or communicated context.

The truth of a $\mathcal{L}_{\mathcal{DROCS}}$ 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 $\exists m \geq n$ s.t. $\forall k \in [n, m)$ $\mathbb{M}, \pi, k \models \varphi$ and $\mathbb{M}, \pi, m \models \psi$,
– $\mathbb{M}, \pi, n \models A\varphi$ iff $\forall \pi' \in B(S,T)$ s.t. $\pi'_{\leq n} = \pi_{\leq n}$, $\mathbb{M}, \pi', n \models \varphi$.

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 selected matching rule instance $\bar{r}$ of 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 may be an arbitrary context which gets overwritten if it is in the agent's dynamic memory $D_M(i)$ or it is a specific context that contradicts with the newly derived context. If agent's memory is full $|V^M(s,i)| = n_M(i)$ then we require that β has to be in $V^M(s,i)$. When the counter value reaches to $n_C(i)$, i cannot perform copy action any more. Furthermore, not all actions are possible in a given state. For example, there may not be any matching rule instance. Note also that only selected matching rule instances can be fired. That is one rule instance may be selected from the conflict set that has the highest priority. If there are multiple rule instances with the same priority, the rule instance to be executed is selected non-deterministically. More formally, we define rule selection strategy as follows:

Definition 2 (Rule selection strategy). *For every state s, agent i, and $r \in V(s,i)$, we say that the rule r matches at state s iff $ant(\bar{r}) \subseteq V(s,i)$ and $cons(\bar{r}) \notin V(s,i)$. Let $\delta : S \times A_g \to \mathcal{G}(\Re)$ be a function that generates matching rule instances of the agent i at state s and $\Re_{mat} \subseteq \mathcal{G}(\Re)$ denote the set of all matching rule instances of the agent i at state s. A set $\Re_{sel}$ is said to be selected rule instances if (i) $\Re_{sel} \subseteq \Re_{mat}$; and (ii) $\forall \bar{r} \in \Re_{sel}\ \nexists \bar{r'} \in \Re_{sel}$ such that $pri(r') \succ pri(r)$.*

Now 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 3 (Available actions). *For every state s and agent i,*

1. *$Rule_{i,r,\beta} \in T_i(s)$ iff $\bar{r} \in \Re_{sel}$, β is a contradictory context (with respect to $cons(\bar{r})$ i.e., if β is α then $cons(\bar{r})$ is $\sim \alpha$ and vice versa) or $\beta \in \Omega$ 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 4 (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 context-aware reasoners is given below:

Definition 5. *$\mathbb{M}(n_M, n_C)$ 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:

- $ByRule_i(P,m) = \neg B_i P \wedge cp_i^{=m} \wedge \bigvee_{\bar{r}\in\Re_{sel}\wedge cons(\bar{r}))=P}(B_i r \wedge \bigwedge_{Q\in ant(\bar{r})} B_i Q)$. This formula describes a state s where it may make a *Rule* transition and believe context P in the next state, 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^* [22].

A2 $\bigwedge_{\alpha\in D_M(i)} B_i\alpha \rightarrow \neg B_i\beta$ 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 $\bigvee_{n=0,\ldots,n_C(i)} cp_i^{=n}$, 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 $m \neq n$, which states that at any given time the value of the copy counter of agent i is unique

A5 $B_i\alpha \rightarrow \neg B_i \sim \alpha$ for any $\alpha \in S_M(i) \cup D_M(i) \subseteq \Omega$, this axiom states that agent does not believe contradictory contexts,

A6 $B_i r \wedge \bigwedge_{\bar{r}\in\Re_{sel}\wedge P\in ant(\bar{r})} B_i P \wedge cp_i^{=n} \wedge \neg B_i cons(\bar{r}) \rightarrow EX(\, B_i cons(\bar{r}) \wedge cp_i^{=n})$, $i \in A_g$. This axiom describes that if a rule matches and is selected for execution, its consequent belongs to some successor state.

A7 $cp_i^{=m} \wedge \neg B_i\alpha \wedge B_j\alpha \rightarrow EX(B_i\alpha \wedge cp_i^{=m+1})$ 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.

A8 $EX(B_i\alpha \wedge B_i\beta) \rightarrow B_i\alpha \vee B_i\beta$, 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.

A9 $B_i\alpha \rightarrow AXB_i\alpha$ 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.

A10 $EX(B_i\alpha \wedge cp_i^{=m}) \rightarrow B_i\alpha \vee ByRule_i(\alpha,m) \vee ByCopy_i(\, \alpha,m)$ for any $\alpha \in \cup\Omega$. This axiom says that a new belief can only be added by one of the valid reasoning actions.

A11a $start \rightarrow cp_i^{=0}$ for all $i \in A_g$. At the start state, the agent has not performed any *Copy* actions.

A11b $\neg EX\ start$. *start* holds only at the root of the tree.

A12 $B_i r$ where $r \in \Re$ and $i \in A_g$. This axiom tells agent i believes its rules.

A13 $\neg B_i r$ where $r \notin \Re$ and $i \in A_g$. This axiom tells agent i only believes its rules.

A14 $\varphi \rightarrow EX\varphi$, where φ does not contain *start*. This axiom describes an *Idle* transition by all the agents.

A15 $\bigwedge_{i\in A_g} EX(\bigwedge_{\alpha\in\Gamma_i} B_i\alpha \wedge cp_i^{=m_i}) \rightarrow EX \bigwedge_{i\in A_g}(\bigwedge_{\alpha\in\Gamma_i} B_i\alpha \wedge cp_i^{=m_i})$ 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 6. $\mathbb{L}(n_M, n_C)$ *is the logic defined by the axiomatisation* ***A1 - A15****.*

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 [22]. 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. Axiom **A5** assures that an agent does not believe contradictory contexts. In the following, we provide the proof for **A6** and **A7**. The proofs for other axioms are similar.

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 $\mathbb{M}, \pi, n \models B_i r \wedge \bigwedge_{\bar{r} \in \Re_{sel} \wedge P \in ant(\bar{r})} B_i P \wedge cp_i^{=m} \wedge \neg B_i cons(\bar{r})$, for some $r \in \Re$ such that $\bar{r} \in \Re_{sel}$, 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 **A7**. 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 [22]. 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

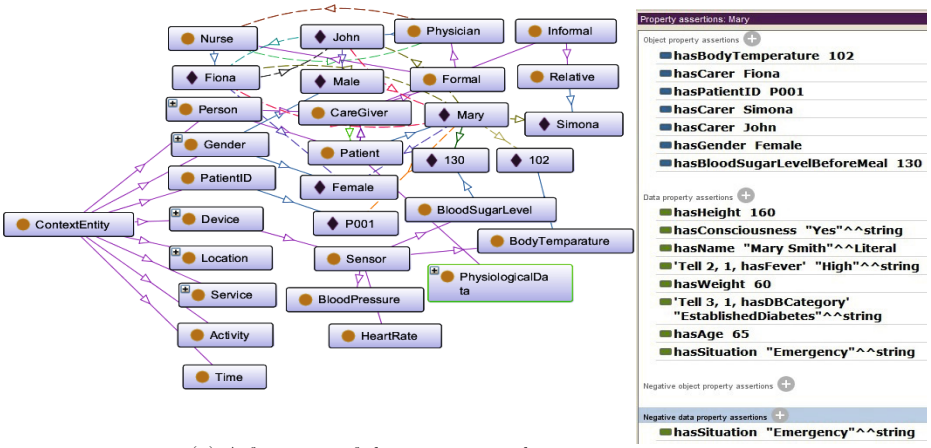
$$\varphi' = \varphi \wedge \bigwedge_{\alpha \in \Omega} (XB_i\alpha \vee \neg XB_i\alpha) \wedge \bigwedge_{n \in \{0 \ldots n_C(i)\}, i \in A_g} (Xcp_i^{=n} \vee \neg Xcp_i^{=n})$$

We then prove that $\mathbb{M}$ is in $\mathbb{M}(n_M, n_C)$ by showing that it satisfies all properties listed in Definition 5. 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

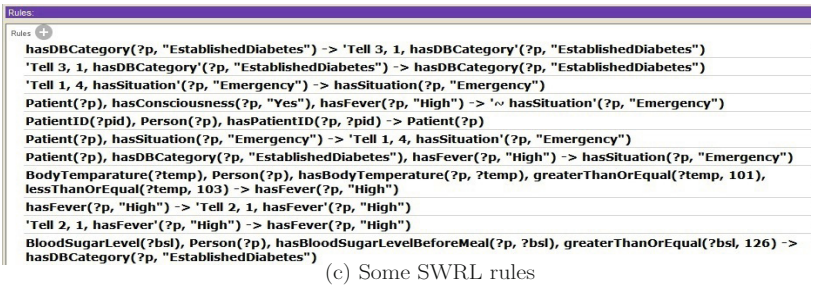
a maximally consistent set (MCS) Γ_0 such that $\Gamma_0 \supseteq V(s_0, i)$ and $start \in \Gamma_0$. Therefore, by axiom **A11**, it is trivial that $cp_i^{=0} \in V(s_0, i)$. We then need to prove that $\forall s \in S$, $act_i \in T_i(s)$, and $i \in A_g$, $\exists s' \in S \cdot sTs'$ and $V(s', i)$ 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 $r \in \Re$ such that $\bar{r} \in \Re_{sel}$. 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 $\bigwedge_{\bar{r} \in \Re_{sel} \wedge P \in ant(\bar{r})} B_i P \wedge cp_i^{=m} \wedge \neg B_i cons(\bar{r}) \in \Gamma$, for some $m \in \{0, \ldots, n_C\}$ and $|V^M(s, i)| \leq n_M(i)$. By axiom **A6** and Modus Ponens (MP), $EX(B_i cons(\bar{r}) \wedge cp_i^{=m}) \in \Gamma$. Therefore, according to the construction, $\exists s' \in S \cdot sTs'$, $V(s', i) \subseteq \Gamma'$ 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 **A7** and axiom **A14**. Then, using axiom **A15** 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 $\forall s, s' \in S \cdot sTs'$, $\exists$ 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 **A8** 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 **A10**, 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 $r \in V(s, i)$ such that $\bar{r} \in \Re_{sel}$, $\alpha = cons(\bar{r}) \notin V(s, i)$. 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 sTs'. Therefore, $\mathbb{M}$ is in $\mathbb{M}(n_M, n_C)$. □

4.1 An Illustrative Example

To illustrate the use of the proposed logical model, let us consider an example system consisting of four agents. A fragment of the context modelling ontology of the system is depicted in Fig. 1 (a). Fig. 1 (b) shows an individualised patient ontology and (c) depicts some SWRL rules. The set of translated rules and initial working memory facts that are distributed to the agents are shown in Table 1, and the goal is to infer the formula $B_4\ hasSituation('Mary, 'Emergency)$ which states that agent 4 believes that the patient Mary has Emergency situation. The reasoning process includes resolving contradictory contextual information. One possible run of the system is shown in Table 2 and Table 3 (continuation). In the tables a newly inferred context at a particular step is shown in blue text. For example, antecedents of rule R11 of agent 1 match the contents of the memory configuration and infers new context $Patient('Mary)$ at step 1. A context which gets overwritten in the next state is shown in red text, and a context which is inferred in the current state and gets overwritten in the next state is shown in magenta text. In the memory configuration, for each agent, left side of the red vertical bar | represents $S_M(i)$ and its right side represents $D_M(i)$. It shows that the size of $D_M(1)$ is 3 units and the size of $D_M(i)$ is 1 unit for all $2 \leq i \leq 4$. We can observe that the resource requirements for the system to derive the goal



(a) A fragment of the context ontology (b) Individualised ptient ontology



(c) Some SWRL rules

Fig. 1. A partial view of the context modelling ontology

formula $B_4\ hasSituation('Mary,'Emergency)$ are 3 messages that need to be exchanged by agent 1 and 1 message that needs to be exchanged by each of the other three agents and 10 time steps. We can also observe that, if we reduce the dynamic memory size for agent 1 by 1, then the system will not be able to achieve the desired goal. We can prove that $X^{10}B_4\ hasSituation('Mary,'Emergency)$ (i.e., from the start state, agent 4 believes $hasSituation('Mary,'Emergency)$ in 10 time steps), where X^{10} is the concatenation of ten LTL next operators X. This is a very simple case; however, if we model a more realistic scenario and increase the problem size, the verification task would be hard to do by hand. Therefore it is more convenient to use an automatic method to verify them, for example using model checking techniques. Due to space constraints, we had to cut this discussion here, however, a $\mathcal{L}_{\mathcal{DROCS}}$ model can be encoded using a standard model checker such as for example the Maude LTL model checker [11] and its interesting resource-bounded properties can be verified automatically.

5 Related Work

In general, rule-based systems have been studied for decades and traditionally rules have been used in theoretical computer science, databases, logic programming, and in particular, in Artificial Intelligence, to describe expert systems,

Table 1. Example rules for a homecare patients' monitoring context-aware system

Agent 1: Patient care
Initial facts: Person('Mary),PatientID('P001), hasPatientID('Mary, 'P001), hasConsciousness('Mary, 'Yes)
R11: Person(?p), hasPatientID(?p, ?pid), PatientID(?pid) → Patient(?p) R12: Tell(2,1, hasFever(?p, 'High)) → hasFever(?p, 'High) R13: Tell(3,1, hasDBCategory(?p, 'EstablishedDiabetes)) → hasDBCategory(?p, 'EstablishedDiabetes) R14: Patient(?p), hasFever(?p, 'High), hasConsciousness(?p, 'Yes)⇒ ∼ hasSituation(?p, 'Emergency) R15: Patient(?p), hasFever(?p, 'High), hasDBCategory(?p, 'EstablishedDiabetes)⇒ hasSituation(?p, 'Emergency) R16: Patient(?p), hasSituation(?p, 'Emergency) → Tell(1,4, hasSituation(?p, 'Emergency))
Rule Priority: R15 ≻ R14
Agent 2: Fever detector
Initial facts: Person('Mary),BodyTemperature('102), hasBodyTemperature('Mary,'102), greaterThanOrEqual('102, '101), lessThanOrEqual ('102, '103)
R21: Person(?p), BodyTemperature(?temp), hasBodyTemperature(?p,?temp), greaterThanOrEqual(?temp, '101), lessThanOrEqual (?temp, '103) → hasFever(?p, 'High) R22: hasFever(?p, 'High)→ Tell(2,1, hasFever(?p, 'High))
Agent 3: Diabetes tester
Initial facts: Person('Mary), BloodSugarLevel('130), hasBloodSugarLevelBeforeMeal('Mary,'130), greaterThan('130,'126)
R31: Person(?p), BloodSugarLevel(?bsl), hasBloodSugarLevelBeforeMeal(?p, ?bsl), greaterThan(?bsl,'126) → hasDBCategory(?p, 'EstablishedDiabetes) R32: hasDBCategory(?p, 'EstablishedDiabetes) → Tell(3,1,hasDBCategory(?p,'EstablishedDiabetes))
Agent 4: Emergency
Initial facts:
R41: Tell(1,4, hasSituation(?p, 'Emergency)) → hasSituation(?p, 'Emergency)

robot behaviour, and behaviour of business. They have found significant application in practice and various researchers have proposed different approaches of defining a knowledge base as a pair of ontology and a set of rules including works by [16,18,23,9]. However, the approaches proposed by [14,13] have mostly influenced the work presented in this paper. In [14] Grosof et. al. have shown that the ontology based modelling techniques can be improved by using the concepts of logic programming. In their work they have noticed certain constraints while translating *DL* axioms into a set of rules. A similar approach proposed by [13] who show that a subset of *DL* languages can be effectively mapped into a set of strict and defeasible rules. Although we follow a similar approach proposed by [14,13] while constructing a set of strict and defeasible rules from an ontology, our purpose and application of those rules are quite different. We use those rules to build a context-aware system as a multi-agent non-monotonic rule-based agents and use a distributed problem solving approach to see whether agents can infer certain contexts while they are resource-bounded. In [21], it has been shown how context-aware systems can be modelled as resource-bounded rule-based systems using ontologies, however it is based on monotonic reasoning where beliefs of an agent cannot be revised based on some contradictory evidence. In [12], OWL ontologies are used to model context-aware systems, the authors exploited classes and properties from ontologies to write rules in Jess

Table 2. (a) One possible run (reasoning) of the system

	Patient care			Fever detector			Diabetes tester			Emergency		
#Steps	Memory Config.1	Action1	# Msg1	Memory Config.2	Action2	#Msg2	Memory Config.3	Action3	#Msg3	Memory Config.4	Action4	#Msg4
0	{*Person*('*Mary*), *PatientID*('*P*001), *hasPatientID*('*Mary*,'*P*001), *hasConsciousness*('*Mary*,'*Yes*) |−, −, −}	−	0	{*Person*('*Mary*), *BodyTemperature*('102), *hasBodyTemperature*('*Mary*,'102), *greaterThanOrEqual*('102,'101), *lessThanOrEqual*('102,'103) |−}	−	0	{*Person*('*Mary*), *BloodSugarLevel*('130), *hasBloodSugarLevelBeforeMeal* ('*Mary*,'130), *greaterThan*('130,'126) |−}	−	0	{|−}	−	0
1	{*Person*('*Mary*), *PatientID*('*P*001), *hasPatientID*('*Mary*,'*P*001), *hasConsciousness*('*Mary*,'*Yes*) |*Patient*('*Mary*), −, −}	*Rule* (R11)	0	{*Person*('*Mary*), *BodyTemperature*('102), *hasBodyTemperature*('*Mary*,'102), *greaterThanOrEqual*('102,'101), *lessThanOrEqual*('102,'103) |*hasFever*('*Mary*,'*High*)}	*Rule* (R21)	0	{*Person*('*Mary*), *BloodSugarLevel*('130), *hasBloodSugarLevelBeforeMeal* ('*Mary*,'130), *greaterThan*('130,'126) |*hasDBCategory*('*Mary*,'*EstablishedDiabetes*)}	*Rule* (R31)	0	{|−}	*Idle*	0
2	{*Person*('*Mary*), *PatientID*('*P*001), *hasPatientID*('*Mary*,'*P*001), *hasConsciousness*('*Mary*,'*Yes*) |*Patient*('*Mary*), −, −}	*Idle*	0	{*Person*('*Mary*), *BodyTemperature*('102), *hasBodyTemperature*('*Mary*,'102), *greaterThanOrEqual*('102,'101), *lessThanOrEqual*('102,'103) |*Tell*(2, 1, *hasFever*('*Mary*,'*High*)}	*Rule* (R22)	1	{*Person*('*Mary*), *BloodSugarLevel*('130), *hasBloodSugarLevelBeforeMeal* ('*Mary*,'130), *greaterThan*('130,'126) |*Tell*(3, 1, *hasDBCategory*('*Mary*,'*EstablishedDiabetes*)}	*Rule* (R32)	1	{|−}	*Idle*	0
3	{*Person*('*Mary*), *PatientID*('*P*001), *hasPatientID*('*Mary*,'*P*001), *hasConsciousness*('*Mary*,'*Yes*) |*Patient*('*Mary*), Tell(2,1, hasFever('Mary, 'High),-}	*Copy*	1	{*Person*('*Mary*), *BodyTemperature*('102), *hasBodyTemperature*('*Mary*,'102), *greaterThanOrEqual*('102,'101), *lessThanOrEqual*('102,'103) |*Tell*(2, 1, *hasFever*('*Mary*,'*High*)}	*Idle*	1	{*Person*('*Mary*), *BloodSugarLevel*('130), *hasBloodSugarLevelBeforeMeal* ('*Mary*,'130), *greaterThan*('130,'126) |*Tell*(3, 1, *hasDBCategory*('*Mary*,'*EstablishedDiabetes*)}	*Idle*	1	{|−}	*Idle*	0
4	{*Person*('*Mary*), *PatientID*('*P*001), *hasPatientID*('*Mary*,'*P*001), *hasConsciousness*('*Mary*,'*Yes*) |*Patient*('*Mary*), Tell(2,1, hasFever('Mary, 'High), Tell(3,1, hasDBCategory('*Mary*,'*EstablishedDiabetes*)}	*Copy*	2	{*Person*('*Mary*), *BodyTemperature*('102), *hasBodyTemperature*('*Mary*,'102), *greaterThanOrEqual*('102,'101), *lessThanOrEqual*('102,'103) |*Tell*(2, 1, *hasFever*('*Mary*,'*High*)}	*Idle*	1	{*Person*('*Mary*), *BloodSugarLevel*('130), *hasBloodSugarLevelBeforeMeal* ('*Mary*,'130), *greaterThan*('130,'126) |*Tell*(3, 1, *hasDBCategory*('*Mary*,'*EstablishedDiabetes*)}	*Idle*	1	{|−}	*Idle*	0
5	{*Person*('*Mary*), *PatientID*('*P*001), *hasPatientID*('*Mary*,'*P*001), *hasConsciousness*('*Mary*,'*Yes*) |*Patient*('*Mary*), hasFever('Mary, 'High), Tell(3,1, hasDBCategory('*Mary*,'*EstablishedDiabetes*)}	*Rule* (R12)	2	{*Person*('*Mary*), *BodyTemperature*('102), *hasBodyTemperature*('*Mary*,'102), *greaterThanOrEqual*('102,'101), *lessThanOrEqual*('102,'103) |*Tell*(2, 1, *hasFever*('*Mary*,'*High*))}	*Idle*	1	{*Person*('*Mary*), *BloodSugarLevel*('130), *hasBloodSugarLevelBeforeMeal* ('*Mary*,'130), *greaterThan*('130,'126) |*Tell*(3, 1, *hasDBCategory*('*Mary*,'*EstablishedDiabetes*)}	*Idle*	1	{|−}	*Idle*	0

Table 3. (b) One possible run (reasoning) of the system

	Patient care			Fever detector			Diabetes tester			Emergency		
#Steps	Memory Config.1	Action1	# Msg1	Memory Config.2	Action2	#Msg2	Memory Config.3	Action3	#Msg3	Memory Config.4	Action4	#Msg4
6	{*Person*('*Mary*), *PatientID*('*P*001), *hasPatientID*('*Mary*,' *P*001), *hasConsciousness*('*Mary*,'*Yes*) ⌊*Patient*('*Mary*), hasFever('Mary, 'High), hasDBCategory('Mary, 'EstablishedDiabetes)}	*Rule* (R13)	2	{*Person*('*Mary*), *BodyTemperature*('102), *hasBodyTemperature*('*Mary*,'102), *greaterThanOrEqual*('102,'101), *lessThanOrEqual*('102,'103) ⌈*Tell*(2, 1, *hasFever*('*Mary*,' *High*))}	*Idle*	1	{*Person*('*Mary*), *BloodSugarLevel*('130), *hasBloodSugarLevelBeforeMeal* ('*Mary*,'130), *greaterThan*('130,'126) ⌈*Tell*(3, 1, *hasDBCategory*('*Mary*,' *EstablishedDiabetes*)}	*Idle*	1	{⌊−}	*Idle*	0
7	{*Person*('*Mary*), *PatientID*('*P*001), *hasPatientID*('*Mary*,' *P*001), *hasConsciousness*('*Mary*,'*Yes*) ⌊*Patient*('*Mary*), hasSituation('Mary, 'Emergency), hasDBCategory('Mary, 'EstablishedDiabetes)}	*Rule* (R15≻R14 Resolving conflicting context)	2	{*Person*('*Mary*), *BodyTemperature*('102), *hasBodyTemperature*('*Mary*,'102), *greaterThanOrEqual*('102,'101), *lessThanOrEqual*('102,'103) ⌈*Tell*(2, 1, *hasFever*('*Mary*,' *High*))}	*Idle*	1	{*Person*('*Mary*), *BloodSugarLevel*('130), *hasBloodSugarLevelBeforeMeal* ('*Mary*,'130), *greaterThan*('130,'126) ⌈*Tell*(3, 1, *hasDBCategory*('*Mary*,' *EstablishedDiabetes*)}	*Idle*	1	{⌊−}	*Idle*	0
8	{*Person*('*Mary*), *PatientID*('*P*001), *hasPatientID*('*Mary*,' *P*001), *hasConsciousness*('*Mary*,'*Yes*) ⌊*Patient*('*Mary*), hasSituation('Mary, 'Emergency), Tell(1,4, hasSituation('Mary, 'Emergency))}	*Rule* (R16)	3	{*Person*('*Mary*), *BodyTemperature*('102), *hasBodyTemperature*('*Mary*,'102), *greaterThanOrEqual*('102,'101), *lessThanOrEqual*('102,'103) ⌈*Tell*(2, 1, *hasFever*('*Mary*,' *High*))}	*Idle*	1	{*Person*('*Mary*), *BloodSugarLevel*('130), *hasBloodSugarLevelBeforeMeal* ('*Mary*,'130), *greaterThan*('130,'126) ⌈*Tell*(3, 1, *hasDBCategory*('*Mary*,' *EstablishedDiabetes*)}	*Idle*	1	{⌊−}	*Idle*	0
9	{*Person*('*Mary*), *PatientID*('*P*001), *hasPatientID*('*Mary*,' *P*001), *hasConsciousness*('*Mary*,'*Yes*) ⌊*Patient*('*Mary*), hasSituation('Mary, 'Emergency), Tell(1,4, hasSituation('Mary, 'Emergency))}	*Idle*	3	{*Person*('*Mary*), *BodyTemperature*('102), *hasBodyTemperature*('*Mary*,'102), *greaterThanOrEqual*('102,'101), *lessThanOrEqual*('102,'103) ⌈*Tell*(2, 1, *hasFever*('*Mary*,' *High*))}	*Idle*	1	{*Person*('*Mary*), *BloodSugarLevel*('130), *hasBloodSugarLevelBeforeMeal* ('*Mary*,'130), *greaterThan*('130,'126) ⌈*Tell*(3, 1, *hasDBCategory*('*Mary*,' *EstablishedDiabetes*)}	*Idle*	1	{⌈*Tell*(1, 4, *hasSituation*('*Mary*,' *Emergency*))}	*Copy*	1
10	{*Person*('*Mary*), *PatientID*('*P*001), *hasPatientID*('*Mary*,' *P*001), *hasConsciousness*('*Mary*,'*Yes*) ⌊*Patient*('*Mary*), hasSituation('Mary, 'Emergency), Tell(1,4, hasSituation('Mary, 'Emergency))}	*Idle*	3	{*Person*('*Mary*), *BodyTemperature*('102), *hasBodyTemperature*('*Mary*,'102), *greaterThanOrEqual*('102,'101), *lessThanOrEqual*('102,'103) ⌈*Tell*(2, 1, *hasFever*('*Mary*,' *High*))}	*Idle*	1	{*Person*('*Mary*), *BloodSugarLevel*('130), *hasBloodSugarLevelBeforeMeal* ('*Mary*,'130), *greaterThan*('130,'126) ⌈*Tell*(3, 1, *hasDBCategory*('*Mary*,' *EstablishedDiabetes*)}	*Idle*	1	{⌊*hasSituation*('*Mary*,' *Emergency*)}	*Infer* (R41)	1

to derive multi-agent rules based system. Thus their modelling part of the system only reflects the static behaviour. In contrast, our ontology-based modelling captures both static and dynamic behaviour of the system using OWL 2 RL and SWRL. A prototype of context management model is presented in [10] that supports collaborative reasoning in a multi-domain pervasive context-aware application. The model facilitates the context reasoning by providing structure for contexts, rules and their semantics. In [5], authors have proposed a distributed algorithm for query evaluation in a Multi-Context Systems framework based on defeasible logic. In their work, contexts are built using defeasible rules, and the proposed algorithm can determine for a given literal P whether P is (not) a logical conclusion of the Multi-Context Systems, or whether it cannot be proved that P is a logical conclusion. However, none of these approaches studies formal specification of context-aware systems considering their resource-boundedness features.

6 Conclusions and Future Work

In this paper, we propose a logical framework for modelling context-aware systems as multi-agent non-monotonic rule-based agents, and the resulting logic $\mathcal{L}_{\mathcal{DROCS}}$ allows us to describe a set of ontology-driven rule-based non-monotonic reasoning agents with bounds on time, memory, and communication. Agents use defeasible reasoning technique to reason with inconsistent information. In future work, we will show how we can encode a $\mathcal{L}_{\mathcal{DROCS}}$ model considering a more realistic system and verify its interesting resource-bounded properties automatically.

References

1. 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)
2. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Transactions on Computational Logic 2(2), 255–287 (2001)
3. Baader, F., McGuinness, D.L., Nardi, D. (eds.): P.F.P.S.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
4. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing 2(4), 263–277 (2007)
5. Bikakis, A., Antoniou, G., Hasapis, P.: Strategies for contextual reasoning with conflicts in ambient intelligence. Knowledge and Information Systems 27(1), 45–84 (2011)
6. Byun, H.E., Chevers, K.: Utilizing context history to provide dynamic adaptations. Applied Artificial Intelligence 18(6), 533–548 (2004)
7. Daniele, L., Costa, P.D., Pires, L.F.: Towards a rule-based approach for context-aware applications. In: Pras, A., van Sinderen, M. (eds.) EUNICE 2007. LNCS, vol. 4606, pp. 33–43. Springer, Heidelberg (2007)
8. Dey, A., Abwowd, G.: Towards a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22, Georgia Institute of Technology

9. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R.: Well-founded semantics for description logic programs in the semantic web. ACM Trans. Comput. Logic 12(2), 1–11 (2011)
10. Ejigu, D., Scuturici, M., Brunie, L.: An ontology-based approach to context modeling and reasoning in pervasive computing. In: PerCom Workshops 2007, pp. 14–19 (2007)
11. 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)
12. 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)
13. Gómez, S.A., Chesñevar, C.I., Simari, G.R.: Inconsistent ontology handling by translating description logics into defeasible logic programming. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 11(35), 11–22 (2007)
14. 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)
15. Horridge, M., Bechhofer, S.: The OWL API: A java API for working with OWL 2 Ontologies. In: 6th OWL Experienced and Directions Workshop (OWLED) (October 2009)
16. Levy, A.Y., Rousset, M.C.: Combining horn rules and description logics in CARIN. Artif. Intell. 104(1-2), 165–209 (1998)
17. Motik, B., Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language: Profiles, W3C Recommendation. (October 2009), `http://www.w3.org/TR/owl2-profiles/`
18. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 3, 41–60 (2005)
19. Pollock, J.L.: Defeasible reasoning. Cognitive Science 11(4), 481–518 (1987)
20. Protégé: The Protégé ontology editor and knowledge-base framework (Version 4.1) (July 2011), `http://protege.stanford.edu/`
21. Rakib, A., Haque, H.M.U., Faruqui, R.U.: A temporal description logic for resource-bounded rule-based context-aware agents. In: Vinh, P.C., Alagar, V., Vassev, E., Khare, A. (eds.) ICCASA 2013. LNICST, vol. 128, pp. 3–14. Springer, Heidelberg (2014)
22. Reynolds, M.: An axiomatization of full computation tree logic. J. Symb. Log. 66(3), 1011–1057 (2001)
23. Rosati, R.: DL+log: Tight integration of description logics and disjunctive datalog. In: Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning, pp. 68–78. AAAI Press (2006)
24. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: Proceedings of the First Workshop on Mobile Computing Systems and Applications, pp. 85–90. IEEE Computer Society, Washington, DC (1994)
25. Schmidt, A., Beigl, M., Gellersen, H.W.: There is more to context than location. Computers and Graphics 23, 893–901 (1998)
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)