Modelling and Reasoning About Context-Aware Agents over Heterogeneous Knowledge Sources

Hafiz Mahfooz Ul Haque, Abdur Rakib(✉), and Ijaz Uddin

School of Computer Science, The University of Nottingham, Malaysia Campus, Semenyih, Malaysia
{khyx2hma,Abdur.Rakib,khyx4iui}@nottingham.edu.my

Abstract. This paper presents a conceptual framework and multi-agent model for context-aware decision support in dynamic smart environments based on heterogeneous knowledge sources. The framework relies on distributed ontologies and allows us to model context-aware agents which reason using rules that are derived from ontologies using the notion of multi-context systems. The use of the proposed framework is illustrated using a simple system developed from ontologies considering three different smart environment domains.

Keywords: Context-aware agents · Multi-context system · Defeasible reasoning · Ontology

1 Introduction

There is no doubt that with an increasing number of smart devices such as smartphones in use, the vast amounts of contextual data being generated has great influence on context-aware mobile computing research. Smartphones have a variety of embedded sensors that can be used to automate data collection and provide a platform to infer rich contextual data about users, including location, time, and environmental condition, among others. This is known as customized information according to the specific context. To be more precise, these sensors can be used to gather the contextual information of a user or to manipulate the context. Different notions of context have been studied across various fields of computer science and various physical and conceptual environmental aspects can be included in the notion of context [11]. Among others, Dey et al. [6] define a context-aware system as a system which uses context to provide relevant information and/or services to its user based on the user's tasks. The formal context modelling and reasoning about context is one of the fundamental research areas in context-aware computing. In the literature, various context modelling and reasoning approaches have been proposed, including ontology and rule-based approach [8,13,14]. In our previous work [13,14], we have developed formal logical frameworks and shown how context-aware systems can be modelled as multi-agent reasoning agents. A formal logical model allows us to capture a system's

P. Cong Vinh et al. (Eds.): ICCASA 2016, LNICST 193, pp. 1–11, 2017.
DOI: 10.1007/978-3-319-56357-2_1

behaviour in a systematic and precise way. This is because a formal logic has simple unambiguous syntax and semantics, which also allows automated reasoning. Our approach to context modelling was based on a domain specific centralised ontology, which allows a formal representation of domain knowledge and advancing contextual knowledge sharing among the agents. However, in a real context-aware deployment setting, we can envisage a coalition of heterogeneous domains which need to mutually share/exchange context knowledge. This needs different modelling approach to deal with distributed context handling considering more than one domain. In this connection, the notion of multi-context systems has been used for interlinking different knowledge sources in order to enhance the expressive capabilities of heterogeneous systems. A multi-context system (MCS) includes a set of contexts and a set of inference rules that allows information to flow among different contexts [7]. In MCS, each context is defined as a self-contained knowledge source which includes the set of axioms and inference rules to model the system and perform local reasoning. Literature highlighted many definitions of multi-context systems (see e.g., [1,5]). In [5], Brewka et al. define multi-context system as a number of people, agents, databases etc. to describe the available information from a set of contexts and inference rules and specify the information flow among these contexts. In [1], Benslimane et al. have described ontology as a context, which is itself an independent self-contained knowledge source having a set of axioms and inference rules with its own reasoner to perform reasoning. In this work, we consider the concept of context in two levels. The first level is based on multi-context system to model heterogeneous systems similar to contextual ontologies studied by [1]. For the second level, we follow the approach proposed in our previous work [13,14], where a context is 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. In this paper, we extend our previous work [13] by introducing a different modelling approach to deal with distributed context handling considering more than one domain. This approach is novel in a sense that context-aware agents use contextual information which are extracted from different knowledge sources.

The rest of the paper is organized as follows. In Sect. 2, we briefly review distributed description logics and related work. In Sect. 3, we contextualize ontologies using three domains to illustrate the central idea of our multi-context systems. In Sect. 4, we briefly describe a tool, *D-Onto-HCR*, which is developed to translate the semantic knowledge into Horn-clause rules which are used to model context-aware systems as multi-agent systems. In Sect. 5, we presents a conceptual framework for modelling context-aware reasoning agents using the MCS notion. In Sect. 6, we illustrate the use of the proposed framework using an example system and conclude in Sect. 7.

2 Background and Related Work

2.1 Distributed Description Logics

Recent developments in the field of semantic web have led to a renewed interest in the distributed knowledge bases [3,9,15]. A growing body of research realizes the significance of extending the OWL based formalisms by providing inter-ontology mappings through distributed description logics. Distributed description logic (DDL) is a formal logical framework which combines different description logics (DLs) knowledge bases to express heterogeneous information. A DDL is basically a generalization of the DL framework, which is designed to formalize multiple ontologies interconnected by semantic mappings [15]. One of the reasons for interconnecting ontologies is to preserve their own identity and specify their independence [9]. DDLs have introduced the notion of multiple ontologies with distributed reasoning where each local ontology has its own local knowledge base. Each local ontology knowledge base consists of TBox and ABox axioms. The correspondences of different ontology axioms is called inter-ontology axioms or bridge rules. Bridge rules map the TBox axioms of one ontology with the TBox axioms of other ontology in an implicit manner. In other words, distributed TBox expresses the semantic relations among local TBoxes via bridge rules. These bridge rules allow concepts of an ontology to subsume a concept from another ontology, and they express the semantic mappings among different ontologies. A bridge rule is an inter-ontology axiom having one of the following forms: $C_i \xrightarrow{\sqsubseteq} D_j$; $C_i \xrightarrow{\sqsupseteq} D_j$; where C_i, D_j are concepts of ontologies O_i and O_j respectively. A distributed DL knowledge base (DKB) is a set of different DL knowledge bases, expressed as a pair $\langle \mathfrak{T}, \mathfrak{A} \rangle$, which consists of distributed TBoxes and ABoxes. Let us assume we have a collection of DLs and each DL is represented by $\{\mathcal{DL}_i\}$, where $i \in I$ is an element of a non empty set of indexes used to identify ontologies.

A distributed TBox (DTBox) defines TBoxes $\{T_i\}_{i \in I}$ of all local DLs from their corresponding domain ontologies, and bridge rules between these TBoxes which are of the form $\mathfrak{B} = \{\mathfrak{b}_{ij}\}$ (which states a set of bridge rules $\mathfrak{B}$ from $\mathcal{DL}_i$ to $\mathcal{DL}_j$ and $\{\forall i, j (i \neq j) \in I\}$). So, DTBox is represented as $\mathfrak{T} = \langle \{T_i\}_{i \in I}, \mathfrak{B} \rangle$.

A distributed ABox (DABox) $\mathfrak{A} = \langle \{A_i\}_{i \in I}, \mathfrak{C} \rangle$ consists of ABoxes $\{A_i\}_{i \in I}$ of all local DLs from their corresponding domain ontologies, and a set of individuals that may either be partial or complete are of the form $\mathfrak{C} = \{\mathfrak{c}_{ij}\}$ which means the individuals corresponds from $\mathcal{DL}_i$ to $\mathcal{DL}_j$ and $\{\forall i, j (i \neq j) \in I\}$.

2.2 Related Work

There has been a renewed research interest in making multiple heterogeneous ontologies interoperate. For example, the work by [15] has introduced a system which can carry out reasoning services with multiple ontologies. The authors have discussed the reasoning problem in multiple ontologies interrelated with semantic mappings, where the results of local reasonings performed in single ontologies are combined via semantic mappings to reason over distributed ontologies. In [4], a framework is presented for multi-context reasoning systems, which allows combining arbitrary monotonic and nonmonotonic logics and non-monotonic bridge rules are used to specify the information flow among contexts. In [2], 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, our purposed approach of reasoning is quite different in a sense that heterogeneous knowledge sources are translated into a set of Horn-clause rules, which are used to model context-aware non-monotonic rule-based agents.

3 Contextualizing Ontologies Using Multi-context System

In [13], we have shown how we use OWL 2 RL ontologies and Semantic Web Rule Language (SWRL) for context-modelling and rule-based reasoning that enables the construction of a formal context-aware system as a distributed non-monotonic rule-based agents. In this work, to model the systems, we extract heterogeneous contextual information from multiple ontologies with the intention of preserving the identity and independence of each specialized domain ontology. To model distributed domains for an example system, we develop three ontologies named as Smart Patient Care (O_{SPC}), Smart Home (O_{SHO}) and Smart Hospital (O_{SHP}) which have their corresponding DL knowledge bases as $\mathcal{DL}_{SPC}$, $\mathcal{DL}_{SHO}$ and $\mathcal{DL}_{SHP}$ respectively. We have discussed how we translate a DL ontology (OWL 2 RL) into a set of plain text Horn-clause rules in [13]. Additionally, we construct the bridge rules which are semantically mapped using distributed DL Knowledge bases. Figure 1 depicts the extracts of class hierarchies of three ontologies. Some of the bridge rules are given below:

$$O_{SPC} : Patient \;\underset{\rightarrow}{\sqsubseteq}\; O_{SHO} : AuthorizedPerson. \quad (1)$$
$$O_{SPC} : Nurse \;\underset{\rightarrow}{\sqsubseteq}\; O_{SHO} : AuthorizedPerson. \quad (2)$$
$$O_{SPC} : Nurse \;\underset{\rightarrow}{\sqsubseteq}\; O_{SHP} : ParamedicalStaff. \quad (3)$$
$$O_{SPC} : CallAmbulance \;\underset{\rightarrow}{\sqsubseteq}\; O_{SHP} : AmbulatoryClinic. \quad (4)$$

Bridge rules 1 and 2 show the relationship between O_{SPC} and O_{SHO}, and rules 3 and 4 show the relationship between O_{SPC} and O_{SHP}. Rule 1 states that a Patient from Patient Care Ontology is an Authorized Person in the Smart Home. Rule 2 and 3 express that a Nurse from the Patient Care Ontology is an Authorized Person in the Smart Home and at the same time a Nurse is a Paramedical staff in the Smart Hospital. These rules can also be represented in first order form as follows:

$$Patient(?p) \mapsto AuthorizedPerson(?p) \quad (1)$$

We model the context using ontologies (including bridge rules, OWL 2 RL and SWRL rules) and extract a set of Horn-clause rules from different ontologies

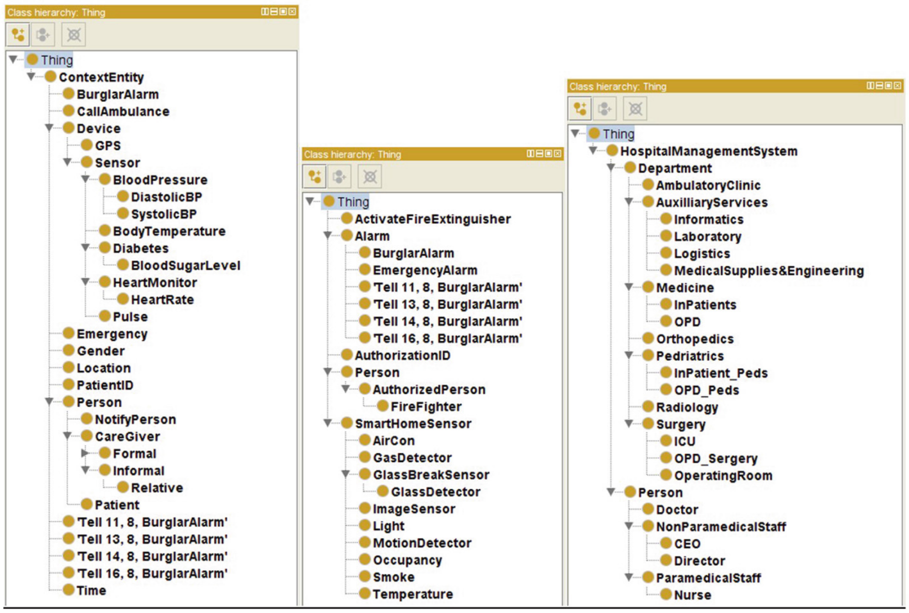

Fig. 1. Class hierarchy of smart environment ontologies

using the tool discussed in the next section. Each agent in the context-aware system has a program, consisting of these extracted Horn clause rules.

4 D-Onto-HCR

To extract the rules from different ontologies, we developed an OWL-API based translator, which takes ontologies as input and then translates the set of axioms (in OWL 2 RL and SWRL form) into a set of plain text Horn-clause rules. The design of the OWL API corresponds to the OWL 2 Structural Specification and this dynamic design model allows developers to provide flexible implementations for major components of the system. In OWL API, the names and hierarchies for the axioms, class expressions and entities correspond to the OWL structural specification. Indeed, there is a proximal one to one translation between OWL API model interfaces and the OWL 2 Structural Specification, implying that this becomes easier to correlate the high level OWL 2 specification with the design of the OWL-API [10]. To extract ontology axioms and facts, we use OWL-API to parse the ontology.

Protégé [12] ontology editor allows SWRL rules to be written in Horn-clause rule format but practically these rules are written in functional syntax which are in DL-Safe rule form. *D-Onto-HCR* translates DL-safe rules axioms into Horn-clause rules format. Additionally, this translator extracts concepts from different ontologies and maps them correspondingly in the from of bridge rules which are

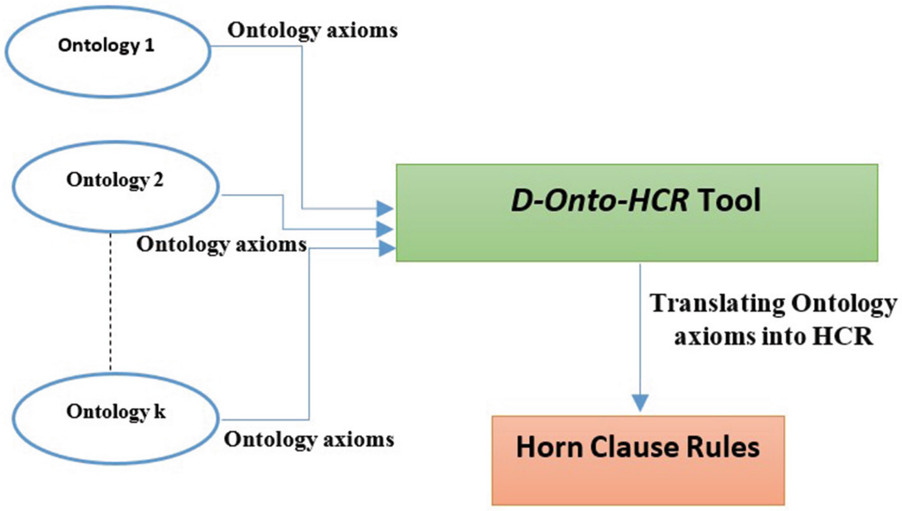

Fig. 2. Distributed semantic knowledge translation process

transformed in OWL 2 RL rule format. These rules are then translated into a set of plain text Horn-clause rules format. Figure 2 shows the distributed semantic knowledge translation process. Each ontology has an ontology IRI (International Resource Identifier) to identify ontology and their classes, properties and individuals. The translation process works as follows: (i) When the tool starts its execution, it loads all listed ontologies from the published source as an input in OWL/XML format; (ii) It uses OWL parser to parse the ontologies into OWL API objects which then extracts the set of TBox and ABox axioms; (iii) The set of TBox axioms are then translated into a set of plain text Horn clause rules; (v) ABox axioms and DL safe SWRL rules are already in the Horn-clause format; (vi) The bridge rules (inter-ontology axioms) are extracted from different ontologies and are also transformed into a set of Horn-clause rules. Multi-context system is a powerful framework for modelling different knowledge sources. Considering the reservations of keeping their own identity and independence as an independent system, the *D-Onto-HCR* tool transforms useful information from these knowledge sources (without making any alteration in ontologies) into a standardized format, i.e., Horn-clause rule format.

5 Multi-agent Model over Heterogeneous Knowledge Sources

We extend the logical framework presented in [13] by incorporating the notion of multi-context systems where rules are derived from heterogeneous semantic knowledge sources. The system consists of $n_{Ag}(\geq 1)$ individual agents $Ag = \{1, 2, ..., n_{Ag}\}$. Each agent $i \in A_g$ has a program, consisting of a finite set of strict, defeasible, and bridge rules, and a working memory, which contains facts. Each agent in the system is represented by a triple $(\Re, \mathcal{F}, \succ)$, where $\mathcal{F}$ is a finite set of facts contained in the working memory, $\Re = (\Re^s, \Re^d, \Re^{br})$ is a finite set of strict, defeasible, and bridge rules, and $\succ$ is a superiority relation on $\Re$. Strict rules ($\Re^s$) are non-contradictory whereas defeasible rules ($\Re^d$) can be defeated based on contrary evidence. Bridge rules ($\Re^{br}$) are non-contradictory rules which represent the distributed knowledge base concepts. In this framework, each context-aware agent is designed to solve a specific problem. Agents in the system acquire contextual information from domain specific ontologies (rules and facts of an agent can be derived from one or multiple ontologies), perform reasoning (based on the information they have in their knowledge bases), communicate with each other, and adapt the system behaviour accordingly. An example set of Horn-clause rules and facts are shown in Table 1. As system moves, the matching rules will be fired based on their predefined priorities which are set by the system designer. That is, 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 represent overall behaviour of the system.

In this framework context-aware agents are modelled using different knowledge sources, where each of them has its own knowledge source and a reasoning strategy. For example, Fig. 3 shows that working memories of three agents contain facts (elements of ABox) from one ontology or multiple ontologies. Agent 1's working memory contains the contextual information C_{11}, C_{12}, C_{15}, and C_{17} which are instances of the Smart Home ontology and C_{22} which is an instance of the Smart Hospital ontology. The working memory of agent 2 has contextual information only from Smart Hospital ontology whereas the working memory of agent N contains the instances from all the ontologies. In a similar fashion bridge rules of an agent include concepts from multiple ontologies.

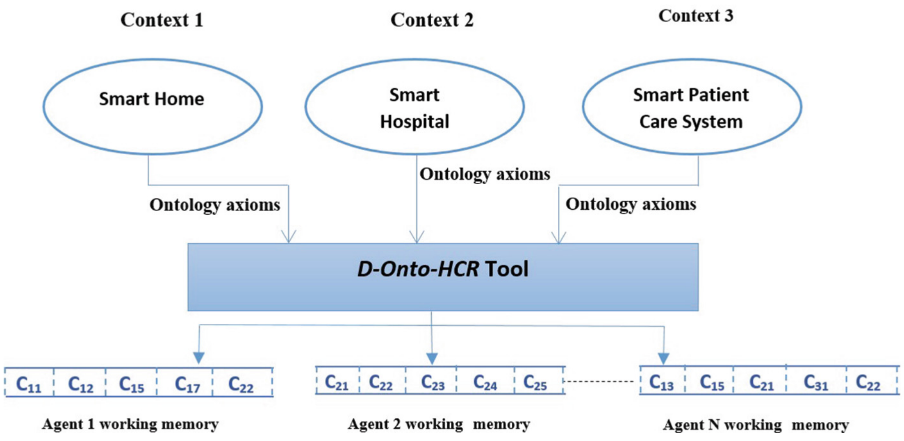

Fig. 3. MCS based context-awareness in the working memory of agent i

Table 1. Example rules for smart environment context-aware system

Agent 1: Home care
Initial facts: Person('John), AuthorizationID('P0001), hasAuthorizationID('John, 'P0001), FireFighter('Simon)
R11: Person(?p), hasAuthorizationID(?p, ?aid), AuthorizationID(?aid) $\rightarrow$ AuthorizedPerson(?p)
R12: FireFighter(?ff) $\mapsto$ AuthorizedPerson(?ff)
R13: Tell(3,1, NotifyPerson(?p, ?loc)) $\rightarrow$ NotifyPerson(?p, ?loc)
R14: NotifyPerson(?p, ?loc), FireFighter(?ff) $\rightarrow$ isRescuedBy(?p, ?ff)
Agent 2: Smoke detector
Initial facts: Smoke('True), hasNotifiedSmokeLocation('True, 'Kitchen)
R21: Smoke(?s), hasNotifiedSmokeLocation(?s, ?loc) $\Rightarrow$ BurglarAlarm(?loc)
R22: Smoke(?s), hasNotifiedSmokeLocation(?s, ?loc) $\Rightarrow \sim$ BurglarAlarm(?loc)
R23: BurglarAlarm(?loc) $\rightarrow$ Tell(2, 3, BurglarAlarm(?loc))
Rule Priority: R21 $\succ$ R22
Agent 3: Emergency monitor
Initial facts: PersonWithinRange('John, 'Kitchen), isFireExtinguisherInstalled ('FEK01, 'Yes)
R31: Tell(2, 3, BurglarAlarm(?loc)) $\rightarrow$ BurglarAlarm(?loc)
R32: BurglarAlarm(?loc) $\rightarrow$ hasAlarmingSituation(?loc, 'Emergency)
R33: hasAlarmingSituation(?loc, 'Emergency), isFireExtinguisherInstalled (?fe, 'Yes) $\rightarrow$ ActivateFireExtinguisher(?loc)
R34: hasAlarmingSituation(?loc, 'Emergency), PersonWithinRange(?p, ?loc) $\rightarrow$ NotifyPerson(?p, ?loc)
R35: NotifyPerson(?p, ?loc) $\rightarrow$ Tell(3,1, NotifyPerson(?p, ?loc))

6 Case Study: Smart Environment Facilitator

We model a smart environment facilitator system considering three different and independent domains, namely Smart Home, Smart Hospital, and Smart Patient Care. The purpose is to model context-aware reasoning agents in healthcare environments which require sharing of knowledge across the domains, including data generated by embedded sensors and wearable smart badges in that environments, while dealing with semantic heterogeneity that exists across the knowledge sources. The Smart Home ontology models the assisted living environment with user-friendly, comfortable and security related facilities. The Smart Hospital ontology models medical services provided to the inpatient and outpatient care. The Smart Patient Care ontology models various devices connected with a patient which monitor the patient's vital information, including blood pressure, blood sugar, and heart rate. As we have already developed ontologies of these domains, to illustrate the use of the framework we consider a very simple

Table 2. Example reasoning steps of the smart environment system

	Home care (Agent 1)			Smoke detector (Agent 2)			Emergency monitor (Agent 3)		
#Steps	Memory Config.1	Action1	# Msg1	Memory Config.2	Action2	#Msg2	Memory Config.3	Action3	#Msg3
0	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid -\}$	–	0	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid -\}$	–	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid -, -\}$	–	0
1	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid AuthorizedPerson('John)\}$	Rule (R11)	0	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid BurglarAlarm('Kitchen)\}$	*Rule* (R21)	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid -, -\}$	Idle	0
2	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid AuthorizedPerson('Simon)\}$	Rule (R12)	0	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid Tell(2, 3, BurglarAlarm('Kitchen))\}$	*Rule* (R23)	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid -, -\}$	Idle	0
3	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid AuthorizedPerson('Simon)\}$	Idle	0	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid Tell(2, 3, BurglarAlarm('Kitchen))\}$	Idle	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid Tell(2, 3, BurglarAlarm('Kitchen)), -\}$	Copy	1
4	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid AuthorizedPerson('Simon)\}$	Idle	0	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid Tell(2, 3, BurglarAlarm('Kitchen))\}$	Idle	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid Tell(2, 3, BurglarAlarm('Kitchen)), BurglarAlarm('Kitchen)\}$	Rule (R31)	1
5	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid AuthorizedPerson('Simon)\}$	Idle	0	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid Tell(2, 3, BurglarAlarm('Kitchen))\}$	Idle	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid Tell(2, 3, BurglarAlarm('Kitchen)), hasAlarmingSituation('Kitchen,' Emergency)\}$	Rule (R32)	1
6	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid AuthorizedPerson('Simon)\}$	Idle	0	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid Tell(2, 3, BurglarAlarm('Kitchen))\}$	Idle	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid ActivateFireExtinguisher('Kitchen), hasAlarmingSituation('Kitchen,' Emergency)\}$	Rule (R33)	1
7	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid AuthorizedPerson('Simon)\}$	Idle	0	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid Tell(2, 3, BurglarAlarm('Kitchen))\}$	Idle	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid NotifyPerson('John,' Kitchen), hasAlarmingSituation('Kitchen,' Emergency)\}$	Rule (R34)	1
8	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid AuthorizedPerson('Simon)\}$	Idle	0	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid Tell(2, 3, BurglarAlarm('Kitchen))\}$	Idle	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid Tell(3, 1, NotifyPerson('John,' Kitchen)), hasAlarmingSituation('Kitchen,' Emergency)\}$	Rule (R35)	1
9	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid Tell(3, 1, NotifyPerson('John,' Kitchen))\}$	Copy	1	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid Tell(2, 3, BurglarAlarm('Kitchen))\}$	Idle	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid Tell(3, 1, NotifyPerson('John,' Kitchen)), hasAlarmingSituation('Kitchen,' Emergency)\}$	Idle	1
10	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid NotifyPerson('John,' Kitchen))\}$	Rule (R13)	1	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid Tell(2, 3, BurglarAlarm('Kitchen))\}$	Idle	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid Tell(3, 1, NotifyPerson('John,' Kitchen)), hasAlarmingSituation('Kitchen,' Emergency)\}$	Idle	1
11	$\{Person('John), AuthorizationID('P0001), hasAuthorizationID('John,' P0001), FireFighter('Simon) \mid isRescuedBy('John,' Simon)\}$	Rule (R14)	1	$\{Smoke('True), hasNotifiedSmokeLocation('True,' Kitchen) \mid Tell(2, 3, BurglarAlarm('Kitchen))\}$	Idle	0	$\{PersonWithinRange('John,' Kitchen), isFireExtinguisherInstalled('FEK01,' Yes) \mid Tell(3, 1, NotifyPerson('John,' Kitchen)), hasAlarmingSituation('Kitchen,' Emergency)\}$	Idle	1

example scenario which includes three agents. As shown in Table 1, each agent has a set of facts and a set of Horn-clause rules. In Table 2, we have shown example reasoning steps to demonstrate how the system generates the specified goals within n time steps and interaction is performed between agents by exchanging messages using *Copy* action [13]. The agents are resource-bounded in terms of the computational time, communication, and space in memory that it consumes [13]. The semantics of the agents' language is based on transition systems and follow the approach of [13]. We view the process of producing new contexts from existing contexts as a sequence of states of an agent, starting from an initial state, and producing the next state by one of the following actions: **Rule:** firing a matching rule instance in the current state (possibly overwriting a context from the previous configuration due to space bound and conflicting contexts appearing in the memory); **Copy:** if agent i has an $Ask(i, j, P)$ (or a $Tell(i, j, P)$) in its current state, then agent j can copy it to its next state provided j's communication counter has not exceeded communication counter threshold value (possibly overwriting a context from the previous configuration due to space bound and conflicting contexts appearing in the memory); and **Idle:** which leaves its configuration unchanged.

As we see the structure of the table, the left most column represents the time step, the rest each three columns *Memory config.*, *Action*, and *#Msg* are assigned for each of the agents which represent the newly inferred contextual information, an appropriate action performed by an agent in a particular step to make a transition from one configuration (state) to another, and the number of messages exchanged respectively. In the column *Memory config.*, the left side of the vertical bar (|) is known as static memory which holds the set of initial facts while the right side (known as dynamic memory) shows the newly derived/communicated contextual information. The size of static memory is determined by the set of initial facts, and the size of dynamic memory is set by the system designer considering minimal memory units required to achieve the desired goals. At the initial configuration, the dynamic memory size of agent 1 and agent 2 is 1 unit while the dynamic memory size of agent 3 is 2 units. However, if we reduce these memory units, the system would not be able to produce a desired goal, e.g., a resident in the smart home is rescued by a fire fighter in case of an emergency alarming situation is reported in the kitchen (i.e., the context $isRescuedBy('John,' Simon)$ appearing in the working memory of one of the agents in the system).

7 Conclusion and Future Work

In this paper, we present a context-aware multi-agent model using the notion of multi-context systems. The proposed framework considers distributed description logic approach to suitably model the core notion of multi-context systems using semantic knowledge sharing. In future work, we aim to develop an optimized Android based application incorporating multi-context system with the specialized domain ontologies having their own conceptualization structure and reasoning strategy.

References

1. Benslimane, D., Arara, A., Falquet, G., Maamar, Z., Thiran, P., Gargouri, F.: Contextual ontologies. In: Yakhno, T., Neuhold, E.J. (eds.) ADVIS 2006. LNCS, vol. 4243, pp. 168–176. Springer, Heidelberg (2006). doi:10.1007/11890393_18
2. Bikakis, A., Antoniou, G., Hasapis, P.: Strategies for contextual reasoning with conflicts in ambient intelligence. Knowl. Inf. Syst. **27**(1), 45–84 (2011)
3. Borgida, A., Serafini, L.: Distributed description logics: directed domain correspondences in federated information sources. In: Meersman, R., Tari, Z. (eds.) OTM 2002. LNCS, vol. 2519, pp. 36–53. Springer, Heidelberg (2002). doi:10.1007/3-540-36124-3_3
4. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, pp. 385–390. AAAI Press (2007)
5. Brewka, G., Roelofsen, F., Serafini, L.: Contextual default reasoning. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence, pp. 268–273 (2007)
6. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**(1), 4–7 (2001). doi:10.1007/s007790170019
7. Eiter, T., Fink, M., Schüller, P., Weinzierl, A.: Finding explanations of inconsistency in multi-context systems. Artif. Intell. **216**, 233–274 (2014)
8. Esposito, A., Tarricone, L., Zappatore, M., Catarinucci, L., Colella, R.: A framework for context-aware home-health monitoring. IJAACS **3**(1), 75–91 (2010)
9. Grau, B.C., Parsia, B., Sirin, E.: Working with multiple ontologies on the semantic web. In: McIlraith, S.A., Plexousakis, D., Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 620–634. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30475-3_43
10. Horridge, M., Bechhofer, S.: The OWL API: a Java API for OWL ontologies. Semant. Web **2**(1), 11–21 (2011)
11. Lieberman, H., Selker, T.: Out of context: computer systems that adapt to, and learn from, context. IBM Syst. J. **39**(3–4), 617–632 (2000)
12. Protégé: The Protégé ontology editor and knowledge-base framework (Version 4.1), July 2011. http://protege.stanford.edu/
13. Rakib, A., Haque, H.M.U.: A logic for context-aware non-monotonic reasoning agents. In: Gelbukh, A., Espinoza, F.C., Galicia-Haro, S.N. (eds.) MICAI 2014. LNCS (LNAI), vol. 8856, pp. 453–471. Springer, Cham (2014). doi:10.1007/978-3-319-13647-9_41
14. Rakib, A., Ul Haque, H.M., 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. LNICSSITE, vol. 128, pp. 3–14. Springer, Cham (2014). doi:10.1007/978-3-319-05939-6_1
15. Serafini, L., Tamilin, A.: DRAGO: distributed reasoning architecture for the semantic web. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 361–376. Springer, Heidelberg (2005). doi:10.1007/11431053_25