UNIVERSITY
OF TRENTO

DIPARTIMENTO DI INGEGNERIA E SCIENZA DELL'INFORMAZIONE

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.disi.unitn.it

BUILDING HETEROGENEOUS MULTI-CONTEXT SYSTEMS BY SEMANTIC BINDINGS

Yuting Zhao, Kewen Wang, Rodney Topor, Jeff Z. Pan and Fausto Giunchiglia

August 2009

Technical Report # DISI-09-043

Note: in Proceedings of The 3rd Chinese Semantic Web Symposium, CSWS09, Nanjing, China, September 2009.

Building Heterogeneous Multi-context Systems by Semantic Bindings

Yuting Zhao[1], Kewen Wang[2], Rodney Topor[2],
Jeff Z. Pan[1], and Fausto Giunchiglia[3]

[1] University of Aberdeen, UK
[2] Griffith University, Australia
[3] University of Trento, ITALY

Abstract. We propose a framework for heterogeneous multi-context systems, in which a special kind of semantic/implicit bridge rules are introduced. Traditional bridge rules in heterogeneous multi-context systems may make the syntax and the semantics of a context more complex, e.g., in the approach of [3] an agent may have to facing a context composed by a description logic systems and a logic program with default negations. In this paper we hide the bridge rules by semantic binding on foreign knowledge fragment, and track the semantic property of a belief/knowledge in one context by a mirror-image of it in the other context. This framework can manage heterogeneous multi-contexts in a simple way, and it keeps the original reasoning properties of the context so that the original reasoning tools are still useful.

1 Introduction

The Semantic Web is a distributed framework that allows software agents conveniently and effectively interpret and apply the data/knowledge that is available on the Web. Nowadays Web contains different kinds of data/knowledge represented by different approaches. Formalization of heterogeneous multi-context systems has become popular in past years because of its capability of modeling semantic web environment [2, 3].

In general understanding a multi-context systems (MCS) presents a number of contexts (i.e. to a number of people/agents/databases/modules, etc.) which holding private (or contextual) knowledge. It also allows knowledge sharing among these contexts, and manages inter-contextual knowledge flow.

Intuitively, a context holds partial and incomplete belief/knowledge about the objective world. For example, in the ancient Indian story "The Blind Men and the Elephant" [4], we can think every blind holds a context about the elephant. In John Godfrey Saxe's version of this legend the first man believes that the elephant is something very like a wall. The second one believes it is like a spear, the third a snake, the forth a tree, the fifth a fan, and the sixth a rope.

[4] `http://en.wikipedia.org/wiki/Blind_Men_and_an_Elephant`

Another example is the magic-box proposed in [8] in Figure-1, which provides a simple illustration of the basic understanding underlying the MCS framework. There are two observers, Mr.1 and Mr.2, each having a partial viewpoint of a box. The box is "magic" and observers cannot distinguish the depth inside it. To express this information, Mr.1 only uses proposition letters l (there is a ball on the left) and r (there is a ball on the right), while Mr.2 also uses a third proposition letter c (there is a ball in the center).

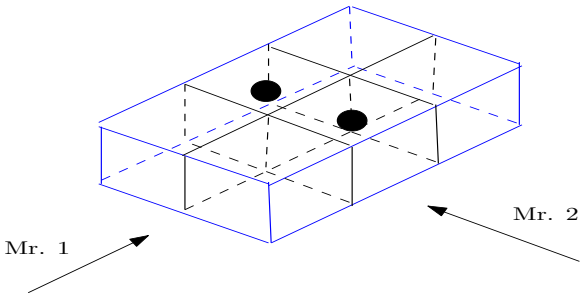

Fig. 1. A magic box.

Researches on formalization of contextual knowledge and reasoning were started from the motivational papers by McCarthy [11, 12] and Giunchiglia [9]. Most of the earlier proposed approaches in this area, including the famous propositional logic of context in [12, 5, 13] and the multi-context systems in [10, 8], are based on classical, monotonic reasoning. Recently the authors of [14] defined a nonmonotonic rule-based MCS framework which contains default negation in the rules. [4] puts forwards a MCS framework based on default logic. It turns out that both of above works are homogeneous, i.e., the inference approaches in different contexts all are the same.

In [3] Brewka and Eiter firstly studied heterogeneous multi-context systems. They proposed a general framework for heterogeneous nonmonotonic MCS reasoning, which is able to combine both monotonic and nonmonotonic logics. For instance, in a multi-context system one context is an ontology presented by a Description Logic system [1], and another is a logic program under Answer Set or Well-founded semantics [7, 6]. In this approach, extended bridge rules which contains default literals are introduced to link different logic systems. According to [3], each context eventually becomes a mixture of two different logic systems; it contains a knowledge base from the original logic (like Description Logic) and a set of bridge rules with default negation. Unfortunately this logic-mix structure of context somehow weakens the significance of this work. On the one hand, it difficult to apply this framework in practical applications, e.g., a WEB agent will have to manage and maintain both a Description Logic systems and an Answer Set Programming systems . On the other hand, techniques of combining two different logic systems are still open problems at the moment.

In this paper we extend the idea of MCS proposed in [3], by replacing bridge rules by semantic bindings. Semantic bindings are different from the bridge rules in the approach of Brewka and Eiter [3]; they do not appear in the syntax of a context, but they effect the semantics of some elements in the language of a context. Semantic binding is able to describe the semantic property of a belief/knowledge in one context by a mirror-image of it in the other context.

One of the advantage of our approach is, it does not make the syntax and the semantics of a context more complex, e.g., if the knowledge base is based on Description Logic then the context is also merely based on Description Logic. The second advantage is, our approach keeps the original reasoning properties of the context, e.g., if the original knowledge is nonmonotonic, then the context in MCS is still nonmonotonic, and the reasoning tools are still useful.

This paper is structured as followings: In Section 2 we introduce the general framework of multi-context systems. In Section 3 and Section 4 we give out the binding semantics, and the propagational binding semantics of multi-context systems respectively. In the following Section we introduce the global semantics of a multi-context system. After we give an example-algorithm for reasoning in a MCS which is based on Description Logic in Section 6, the conclusion is given in Section 7.

2 Multi-context systems

In this section we introduce a general framework of MCS which using semantic bindings instead of bridge rules proposed in [3]. In order to present different logics in different contexts in one MCS, firstly we start from a general definition of a logic proposed in [3].

Definition 1 (Logic). *A* logic $\mathtt{L}$ *is a triple* $(\boldsymbol{KB}_L, \boldsymbol{BS}_L, \boldsymbol{ACC}_L)$ *in which:*

1. $\boldsymbol{KB}_L$ *is the set of well-formed knowledge bases of L. We assume each element of* $\boldsymbol{KB}_L$ *is a set.*
2. $\boldsymbol{BS}_L$ *is a collection of possible belief sets.*
3. $\boldsymbol{ACC}_L$*:* $\boldsymbol{KB}_L \rightarrow 2^{\boldsymbol{BS}_L}$ *is a function describing the "semantics" of the logic by assigning to each element of* $\boldsymbol{KB}_L$ *a set of acceptable sets of beliefs.*

Example 1. Following are examples of logics over a signature Σ:

- Description Logic (DL)
  - **KB**: is the set of pairs of TBox and ABox in Description Logic over Σ.
  - **BS**: is the set of sets of models of Description Logic.
  - **ACC**(kb): is the set of possible models of a Description Logic system kb.
- Normal logic programs under answer set semantics (ASP). (Due to [3])
  - **KB**: is the set of normal logic programs over Σ.
  - **BS**: is the set of sets of atoms over Σ.
  - **ACC**(kb): is the set of kb's answer sets.
- Propositional logic under the closed world assumption (SAT). (Due to [3])

- **KB**: is the set of sets propositional formulas over Σ.
- **BS**: the set of deductively closed sets of propositional Σ-formulas.
- **ACC**(kb): the (singleton set containing the) set of kb's consequences under closed world assumption.

Under common understanding, a *well-formed formula (WFF)* is a legal expression which indicates a complex "meaning" in a logic theory. It could be a symbol or string of symbols (a formula) that is generated by the formal grammar of a formal logic language. Here we do not specify a detailed language for each of the logics we talk about. So when we say a WFF of a logic, we use the formal grammar in its most general sense. For example, we think $\forall Haschild.Female$ as a WFF in Description Logic, but obviously not a WFF in First Order Logic.

Given a logic $L_i = (\mathbf{KB}_i, \mathbf{BS}_i, \mathbf{ACC}_i)$ over a signature Σ, if a knowledge base $kb \in \mathbf{KB}_i$ is true under a belief set M, that is, $M \in \mathbf{ACC}_i(kb)$, then we say M is a *model* of kb. We say a model M *satisfies* a WFF λ if λ is true under $M \in \mathbf{ACC}_i$ according to the semantics of L_i. This fact is denoted by $M \models \lambda$.

MCS allows knowledge sharing among contexts. So a context could use some knowledge fragments of some other contexts. Here we need a double-index way to show (1) where a knowledge fragment is using, and (2) where it is from. For example, we use $(1:2:raining)$ to present a belief "raining" in context C_1, but originally it comes from a WFF in context C_2. Intuitively it can be interpreted as "C_1 says that C_2 tells her it is raining now". For a double-index denotation $(i:j:B)$ in a MCS where $i \neq j$, we call it the *foreign entity* of context C_i. Because its prefix "$i:j$ " indicates originally B is from context C_j, $(i:j:B)$ is also called a *j-entity*. As for a double-index denotation like $(i:j:B)$ where $i = j$, We call it *local entity* in context C_i. In our understanding a local entity is actually a normal WFF in a context. So we always use the convenience to skip the double-index prefix and just write B.

We note in this paper the double-index denotation of foreign entity does not extend the language of a logic by introducing new language constructors from some other logics. The denotation that double-index prefixed to some WFF in another context (maybe also in another logic) is just treated as an normal atomic symbol in the current context. For example, a foreign entity $(i:j:\exists hasChild.Male)$ in context C_i under Description Logic is different from $\exists(i:j:hasChild).(i:j:Male)$. The whole string of the former is an atomic concept and the "$\exists$" in it is not an existential quantification; the latter is a complex concept sentence which is composed by an existential quantification "$\exists$", a foreign role "$(i:j:hasChild)$", and another foreign concept "$(i:j:Male)$". So a double-index prefixed WFF in Description Logic like $(i:j:\exists hasChild.Male)$ can appear as an atom in Logic programming, or a double-index prefixed WFF in First Order Logic like $(i:j:\exists xA(x))$ can be used as an atom in Description Logic.

Suppose context C_i has a j-entity λ. Obviously in λ there is a prefix "$i:j$ ". If we change the prefix in λ from "$i:j$ " to "$j:j$ ", and then we get λ'. We call λ' the *original image* of λ. Actually λ' must belong to C_j, it is just the WFF of logic L_j, and we can skip the prefix "$j:j$ " for convenience.

For example, suppose context C_1 is based on First Order Logic, and C_2 Description Logic. C_2 has a entity $\lambda_2 = Person \sqcap \forall hasChild.Female$. If there is a 2-entity $\lambda_1 = (1:2:Person \sqcap \forall hasChild.Female)$ in context C_1, then we say λ_2 is the original image of λ_1.

Intuitively the double-index denotation indicates that a foreign entity is bound to its original image. For example, when we use a foreign entity $(i:j:\exists hasChild.Male)$ in context C_i in reasoning, we always want to use the meaning of $\exists hasChild.Male$ in context C_j. In another words, a knowledge fragment $\exists hasChild.Male$ in context C_j is shared by context C_i via the foreign entity $(i{:}j{:}\exists hasChild.Male)$. Details of the semantics will be given in next section.

Definition 2 (Multi-context systems). *A* multi-context systems (MCS) $M = (C_1, \cdots, C_n)$ *is a collection of contexts* $C_i = (L_i, kb_i)$ *where* $L_i = (\boldsymbol{KB}_i, \boldsymbol{BS}_i, \boldsymbol{ACC}_i)$ *is a logic and* $kb_i \in \boldsymbol{KB}_i$ *is a knowledge base.*

Comparing with the definition-3 in [3] we note that in our framework the set of bridge rule is invisible. Actually every foreign entity in one context acts as a kind of special bridge rule, like

$$p \leftarrow (j:p),$$

which supports the information flow from another context. That is the reason why we declare that in our framework the "bridge rule", if we have to use this term, is "implicate". From this aspect the MCS framework in our approach is a special case of the one proposed in [3]. But actually our approach is more practical in operating, and the semantics is simple and easy to understand.

In this paper we assume all MCS are *closed*, in the sense that, if there is a foreign entity $(i{:}j{:}p)$ belongs to C_i and $C_i \in M$, then we must have $C_j \in M$.

3 Binding semantics of multi-context systems

In this section we show the reason why we declare the bridge rule in this framework is "semantic". In syntax a foreign entity is different from the original one because it is prefixed by the index number of current user (context). In order to activate the knowledge flow among contexts, we introduce the *binding semantics* to foreign entity. In this way a foreign entity in one context can stimulate the meaning of a belief/knowledge in another context by tracking its semantic actions/properties. It become something like a mirror-image of the original one.

Definition 3 (Local semantics). *Let* $C_i = (L_i, kb_i)$ *be a context, where* $L_i = (\boldsymbol{KB}_i, \boldsymbol{BS}_i, \boldsymbol{ACC}_i)$ *is a logic and* $kb_i \in \boldsymbol{KB}_i$. *We say a WFF* λ *is a* local belief *of* C_i, *iff there is a model* $M_k \in \boldsymbol{ACC}_i(kb_i)$, *and we have* M_k *satisfies* λ. *This fact is denoted by* $C_i \models_L \lambda$.

Definition 4 (Binding consistent semantics). *Let* $C_i = (L_i, kb_i)$ *be a context, where* $L_i = (\boldsymbol{KB}_i, \boldsymbol{BS}_i, \boldsymbol{ACC}_i)$ *is a logic and* $kb_i \in \boldsymbol{KB}_i$. *For any* $M \in$

ACC$_i(kb_i)$*, we say* M *is a* binding consistent model *of* C_i *iff, for every* j*-entity* λ *which has an original image* λ'*, we have*

$$M \models \lambda \text{ iff } C_j \models_L \lambda',$$

This fact is denoted by $M \models_B \lambda$.

If there is a WFF λ *s.t. for a binding consistent model* M_k, $M_k \models_B \lambda$*, then we say* λ *is a* binding consistent belief *of* C_i*. This fact is denoted by* $C_i \models_B \lambda$.

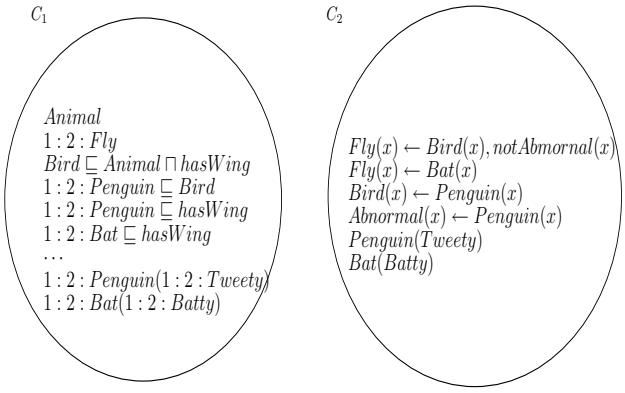

Fig. 2. A multi-context example.

Example 2. In the MCS in Figure-2, contexts C_1 and C_2 are based on Description Logic and Answer Set programming respectively[5]. Obviously C_1 itself could not answer "Does Tweety fly?" or something like that. In C_2 the only answer set is {*Fly(Batty), Bat(Batty), Penguin(Tweety), Abnormal(Tweety), Bird(Tweety)* }. So when C_1 is attempting to answer "Does Batty fly?" , it finds out that $C_2 \models Fly(batty)$, then it can answer "Yes!" under the binding consistent semantics.

When C_1 is attempting to answer "Does Tweety fly?" , it finds out that $C_2 \models \neg Fly(Tweety)$, then it can conclude $C_1 \models (1:2:Fly)((1:2:Tweety))$ under the binding consistent semantics.

Obviously under the local semantics a foreign entity is treated as local one. In another words, the semantics of foreign entity does not constrained by its original image under the local semantics. While under the binding consistent semantics a foreign entity is also constrained by its original image. So only a subset of local models can be binding consistent model.

[5] We note in Answer Set Programming literals with variables, like Fly(x) in C_2 in Figure-2, are called *schematic literals*, in which the variable stands for an unspecified individual.

In this paper, we use $\Pi_{\mathcal{B}}(C_i) = \{\lambda |\ C_i \models_B \lambda\}$ to denote the set of binding consistent belief of C_i. Comparably, we also use $\Pi_{\mathcal{L}}(C_i) = \{\lambda |\ C_i \models_L \lambda\}$ to denote the set of local entity set of C_i.

Theorem 1. *For a context C_i, we have $\Pi_{\mathcal{L}}(C_i) \subseteq \Pi_{\mathcal{B}}(C_i)$.*

Proof. According to Definition-3 and definition-4, the proof is straightforward.

4 Propagational binding semantics

As shown in above section, under the binding semantics information flow only makes one step; it does not propagate away to the third context. In this section we introduce a semantics which allows the propagation of binding semantics in the whole MCS.

Definition 5 (Propagational binding consistent semantics). *Let $C_i = (L_i, kb_i)$ be a context, where $L_i = (\boldsymbol{KB}_i, \boldsymbol{BS}_i, \boldsymbol{ACC}_i)$ is a logic and $kb_i \in \boldsymbol{KB}_i$. For any $M \in \boldsymbol{ACC}_i(kb_i)$, we say M is a* propagational binding consistent model *of C_i if, for every* j*-entity λ in C_i, let λ' be the original image of λ, we have $M \models \lambda$ if*

1. $C_j \models_L \lambda'$, *or*
2. $C_j \models_B \lambda'$, *or*
3. $C_j \models_P \lambda'$

This fact is denoted by $M \models_P \lambda$.

If there is a entity λ s.t. for every propagational binding consistent model M_k, $M_k \models_P \lambda$, then we say λ is a propagational binding consistent belief *of C_i. This fact is denoted by $C_i \models_P \lambda$.*

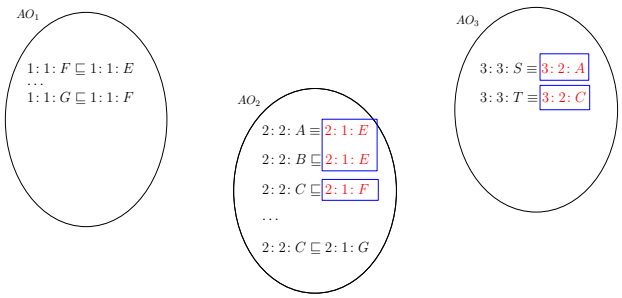

Fig. 3. Propagation in MCS.

Example 3. The MCS in Figure-3 contains three contexts. We can not have $C_3 \models_P (3{:}3{:}T) \sqsubseteq (3{:}3{:}S)$ under the binding semantics, but we can conclude $C_3 \models_P (3{:}3{:}T) \sqsubseteq (3{:}3{:}S)$ under the propagational binding semantics.

$$\frac{\begin{array}{l}\because C_1 \models_L (1{:}1{:}F) \sqsubseteq (1{:}1{:}E),\\ \therefore C_2 \models_B (2{:}1{:}F) \sqsubseteq (2{:}1{:}E)\\ \because C_2 \models_L (2{:}2{:}A) \sqsubseteq (2{:}1{:}E),\\ \quad \textit{and}\ \ C_2 \models_L (2{:}2{:}C) \sqsubseteq (2{:}1{:}F)\\ \therefore C_2 \models_B (2{:}2{:}B) \sqsubseteq (2{:}2{:}A)\\ \therefore C_3 \models_P (3{:}2{:}B) \sqsubseteq (3{:}2{:}A)\\ \because C_3 \models_L (3{:}3{:}S) \equiv (3{:}2{:}A),\\ \quad \textit{and}\ \ C_3 \models_L (3{:}3{:}T) \equiv (3{:}3{:}C)\end{array}}{C_3 \models_P (3{:}3{:}T) \sqsubseteq (3{:}3{:}S)}.$$

So we have $C_3 \models_P (3{:}3{:}T) \sqsubseteq (3{:}3{:}S)$.

In this paper, we use $\Pi_{\mathcal{P}}(C_i) = \{\lambda |\ C_i \models_P \lambda\}$ to denote the set of propagational binding consistent belief of C_i, then,

Theorem 2. *For a context C_i, we have $\Pi_{\mathcal{B}}(C_i) \subseteq \Pi_{\mathcal{P}}(C_i)$.*

Proof. According to Definition-5 and definition-4, the proof is straightforward.

5 Example algorithm: Reasoning in a MCS which is based on Description Logic

In this section, we present a simple tableaux algorithm on Description Logic *ALCN* in order to illuminate how to reasoning in a MCS. This algorithm is designed for verifying class satisfiability in an Description Logic systems, and can also be used to verify class subsumption.

Tableaux algorithms (first by [15]) are very useful to solve class satisfiability problem. They test the satisfiability of a class λ [6] by trying to construct an interpretation for λ, which is represented by a *completion tree* T which is formally defined as following: A *completion tree* contains following elements: x_0 is the root of T , N and E are the sets of nodes and edges, respectively, of T , and labis a function that maps a node x in T to its label Lab(x) , and an edge $\langle x, y\rangle$ in T to its Lab($\langle x, y\rangle$), respectively. A tableaux algorithm starts from an labeled initial tree (usually simply a root node), and is expanded by repeatedly applying the completion rules. The algorithm terminates either when T is *complete* (no further completion rules can be applied) or when an obvious contradiction, or *clash*, has been revealed.

[6] Here we assume λ is in negation normal form; i. e. , negation is only applied to class names.

In this section, we use procedure $\texttt{Tab}(C_k, T_\lambda)$ as a well known (local) *ALCN* tableaux algorithm to expand $\texttt{T}$ *w.r.t.* a local TBox. Given a MCS in which contexts are based on Description Logic, the procedure $\mathcal{B}$-Tab(C_k, T_λ) verifies the satisfiability of an ALC class description λ in context C_k under the binding semantics.

Algorithm A-1: $\mathcal{B}$-Tab(C_k, T_λ)

1: Let $\texttt{T}$:= $Tab(C_k, T_\lambda)$ // local expansion
2: **repeat**
3: **if** $\texttt{T}$ has a clash **then**
4: *return unsatisfiable*
5: **end if**
6: **for** every context $C_i \in$ MCS, $i \neq k$ **do**
7: project i-entity subtree, whose notes are all i-entity, to T_i'
8: $\texttt{T}'_i := Tab(C_i, \texttt{T}'_i)$ // local expansion *w.r.t.* C_i
9: **if** any of $\{\texttt{T}'_i\}$ has a clash **then**
10: **if** $\texttt{T}$ is backtrackable **then**
11: $\texttt{T}$:=Tab(C_i, $\texttt{T}$, *backtrack*) // backtrack and expand
12: **else**
13: *return unsatisfiable*
14: **end if**
15: **end if**
16: **end for**
17: *return satisfiable*
18: **until** false

In this algorithm, $\texttt{T}$ is initialized with a root x_0 with $\texttt{lab}(x_0) = \{\lambda\}$, and is expanded by local completion rules *w.r.t.* C_k (line 1 of **A-1**). As $\texttt{T}$ can have multiple binding neighbor contexts, each of them should be taken care (line 6 of **A-1**). Note that $\texttt{T}$ might not contain i-entity, the algorithm just project the maximal i-entity sub-trees, and then expand them by local completion rules *w.r.t.* C_i, (lines 8 of **A-1**). If any of the projected sub-tree has a clash, $\texttt{T}$ needs to be backtracked (line 11 of **A-1**), expanded and start the checking all over again.

Example 4. Let MCS = $\{C_1, C_2\}$, in which $C_1 = \{(1:1:cat) \sqsubseteq \neg(1:1:dog)\}$, $C_2 = \{(2:2:cat\_Man) \sqsubseteq \forall(2:2:hasPet).(2:1:cat),\ (2:2:dog\_Man) \sqsubseteq \forall(2:2:hasPet).(2:1:dog)\}$. We check if $(2:2:cat\_Man) \sqcap (2:2:dog\_Man)$.

We first build a clash-free completed completion tree $\texttt{T}$ *w.r.t.* C_2, it consists of two connected nodes x_0 and x_1, where x_0 is labeled with $\texttt{Lab}(x_0) = \{(2:2:cat\_Man) \sqcap (2:2:dog\_Man),\ (2:2:cat\_Man),\ (2:2:dog\_Man),\ \forall(2:2:hasPet).(2:1:cat),\ \forall(2:2:hasPet).(2:1:dog)\ \}$, x_1 is labeled with $\texttt{Lab}(x_1) = \{(2:1:dog),\ (2:1:cat)\}$ and the edge $\langle x_0, x_1\rangle$ is labeled with $\texttt{Lab}(\langle x_0, x_1\rangle) = (2:2:hasPet)$. Obvious there are two 1-entity subtree, nodes x_0 and x_1. After projection, $\texttt{T}'_1$ contains node x'_0 and $\texttt{Lab}(x'_0) = \emptyset$, and $\texttt{T}'_2$ contains x'_1 and $\texttt{Lab}(x'_1) = \{(2:1:dog),\ (2:1:cat)\}$. Then we expand $\texttt{T}'_2$ in C_1, and get $\texttt{Lab}(x'_1) = \{(2:1:dog),\ (2:1:cat),\ \neg(1:1:dog)\}$. There is a clash, so $\texttt{T}'_2$ and $\texttt{T}$ are not complete and $(2:2:cat\_Man) \sqcap (2:2:dog\_Man)$ is unsatisfiable

w.r.t. MCS. □

This example shows by "project" operation we can distribute a partial reasoning job related to j-entity to the original j context. It implies that asking queries to other contexts could be a simple way to realize semantic bindings.

6 Conclusion

Intuitively the job of formalization of multi-context systems is to manage semantic cooperation and reasoning among distributed contexts.

In this paper we propose a general framework for heterogeneous multi-context systems which can contain different kind of logic systems. Different from existing works based on syntaxual/explicate bridge rules, we introduce a special kind of semantic/implicate bridge rules to set up semantic cooperation among multi-contexts. We hide the bridge rules by semantic binding on foreign knowledge fragment, and track the semantic property of a belief/knowledge in one context by a mirror-image of it in the other context.

From this aspect, we call these *semantic bridge rules*, and those in above mentioned approaches *syntaxual bridge rule*. The semantic bridge rules is our framework is realized by binding semantics [7], which is able to describe the semantic property of a belief/knowledge in one context by a mirror-image of it in the other context.

This framework can manage heterogeneous multi-contexts but it does not make the syntax of the contextual knowledge base more complex, which is an advantage over the approach of [3]. As the same time it keeps the original reasoning properties of the context so that the original reasoning tools are still useful.

Although we give out an simple algorithm in Section-5, in order to illuminate how this semantic binding idea can be realized in a multi-context systems in which each context is based on Description Logic, our discussion is mainly theoretical. An open future work is related to implementation, say how to generalize Algorithm(A-1) to other logic systems.

References

1. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
2. Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini1, and Heiner Stuckenschmidt. C-owl: Contextualizing ontologies. In *Proceedings of the 2nd International Semantic Web Conference (ISWC2003)*, pages 20 – 23, Sundial Resort, Sanibel Island, Florida, USA., October 2003.

[7] Another version of binding semantics on Description Logic is proposed in [16].

3. Gerhard Brewka and Thomas Eiter. Equilibria in heterogeneous nonmonotonic multi-context systems. In *Proc. AAAI-07*, 2007.
4. Gerhard Brewka, Floris Roelofsen, and Luciano Serafini. Contextual default reasoning. In *Proc. IJCAI-07*, 2007.
5. Sasa Buvač and Ian A. Mason. Propositional logic of context. In Richard Fikes and Wendy Lehnert, editors, *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 412–419, Menlo Park, CA, 1993. AAAI Press.
6. Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):619–649, 1991.
7. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth Bowen, editors, *Proceedings of the Fifth International Conference on Logic Programming*, pages 1070–1080, Cambridge, Massachusetts, 1988. The MIT Press.
8. C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, April 2001.
9. F. Giunchiglia. Contextual reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine*, XVI:345–364, 1993.
10. F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics, or: how we can do without modal logics. *Artificial Intelligence*, 65(1):29–70, 1994.
11. John McCarthy. Generality in artificial intelligence. *Communications of ACM*, 30(12):1030–1035, 1987.
12. John McCarthy. Notes on formalizing contexts. In Tom Kehler and Stan Rosenschein, editors, *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 555–560, Los Altos, California, 1993. Morgan Kaufmann.
13. John McCarthy and Sasa Buvač. Formalizing context (expanded notes). In Sasa Buvač and Łucia Iwańska, editors, *Working Papers of the AAAI Fall Symposium on Context in Knowledge Representation and Natural Language*, pages 99–135, Menlo Park, California, 1997. American Association for Artificial Intelligence.
14. Floris Roelofsen and Luciano Serafini. Minimal and absent information in contexts. In *Proc. IJCAI-05*, 2005.
15. Manfred Schmidt-Schauβ and Gert Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26, 1991.
16. Yuting Zhao, Kewen Wang, Rodney Topor, Jeff Z. Pan, and Fausto Giunchiglia. Semantic cooperation and knowledge reuse by using autonomous ontology. In *(to appear) Proceeding of The 6th International Semantic Web Conference (ISWC'07)*, Busan, Korea, 2007.