Evolving Bridge Rules in Evolving Multi-Context Systems

Ricardo Gonçalves, Matthias Knorr, and João Leite

CENTRIA & Departamento de Informática, Faculdade Ciências e Tecnologia
Universidade Nova de Lisboa, Portugal

Abstract. In open environments, agents need to reason with knowledge from various sources, represented in different languages. Managed Multi-Context Systems (mMCSs) allow for the integration of knowledge from different heterogeneous sources in an effective and modular way, where so-called bridge rules express how information flows between the contexts. The problem is that mMCSs are essentially static as they were not designed to run in a dynamic scenario. Some recent approaches, among them evolving Multi-Context Systems (eMCSs), extend mMCSs by allowing not only the ability to integrate knowledge represented in heterogeneous KR formalisms, but at the same time to both react to, and reason in the presence of commonly temporary dynamic observations, and evolve by incorporating new knowledge. These approaches, however, only consider the dynamics of the knowledge bases, whereas the dynamics of the bridge rules, i.e., the dynamics of how the information flows, is neglected. In this paper, we fill this gap by building upon the framework of eMCSs by further extending it with the ability to update the bridge rules of each context taking into account an incoming stream of observed bridge rules. We show that several desirable properties are satisfied in our framework, and that the important problem of consistency management can be dealt with in our framework.

1 Introduction

In *Open Multi-Agent Systems*, the paradigm for knowledge representation and reasoning (KRR) is rapidly changing from one where each agent has its own monolithic knowledge base written in some language into one where each agent has to deal with several external heterogeneous sources of knowledge, possibly written in different languages (see, e.g.,[1,22,25] and references therein). These sources of knowledge include the large number of available ontologies and rule sets, as well as the norms and policies published by the *institutions*, the information communicated by other agents, to name only a few.

Each agent needs to be able to deal with such distributed sources of knowledge, taking into account the interactions and possible flows of information between them. For example, the agent may use inferences drawn from some ontology to justify the conclusions drawn from some rules in another knowledge base; or the agent may use some piece of information published by some other agent to infer that some action it is about to undertake will not violate some norm published by some institution. Unlike approaches that aim to integrate several knowledge bases to obtain a common view of the system, our focus is on how a particular agent can integrate several knowledge bases to obtain its own view of the system. One consequence of our focus is that no coordination between agents is involved since the way knowledge bases are combined, how they interact, and how information flows between them, is internal, and ultimately private, to the agent in question.

Two common ways of integrating heterogeneous knowledge exist, namely either relying on hybrid languages (e.g., [20,27], and [24] with its reasoner NoHR [23]), to which other languages can be translated, or modular approaches (e.g., [9,14]) in which different formalisms and knowledge bases are considered as modules, and means are provided to model the flow of information between them. Among the latter, Multi-Context Systems (MCSs) [9,19,28] are particularly general and have gained some attention by agent developers [7,12,29].

MCSs consist of a set of contexts, each of which is a knowledge base in some KR formalism, such that each context can access information from other contexts using so-called bridge rules. Such non-monotonic bridge rules add their head to the context's knowledge base provided the queries (to other contexts) in the bodies are successful. Managed Multi-Context Systems (mMCSs) were introduced in [10] to extend MCSs by allowing operations, other than simple addition, to appear in the heads of bridge rules. This allows mMCSs to properly deal with the problem of consistency management within contexts.

A recent challenge for KR languages is the shift from static scenarios which assume a one-shot computation, usually triggered by a user query, to open and dynamic scenarios where there is a need to react and evolve in the presence of incoming information. Examples include EVOLP [2], Reactive ASP [17,16], C-SPARQL [6], Ontology Streams [26] and ETALIS [4], to name only a few.

Whereas mMCSs are quite general and flexible to address the problem of integration of different KR formalisms, they are essentially static in the sense that the contexts do not evolve to incorporate the changes in the dynamic scenarios. In such scenarios, new knowledge and information is dynamically produced, often from several different sources – for example a stream of raw data produced by some sensors, new ontological axioms written by some user, newly found exceptions to some general rule, etc.

To address this issue, two recent frameworks, evolving Multi-Context Systems (eMCSs) [21] and reactive Multi-Context Systems (rMCSs) [8,15,11] have been proposed sharing the broad motivation of designing general and flexible frameworks inheriting from mMCSs the ability to integrate and manage knowledge represented in heterogeneous KR formalisms, and at the same time be able to incorporate knowledge obtained from dynamic observations.

Whereas some differences set eMCSs and rMCSs apart, namely regarding how observations are handled, and the kind of state transitions that can be made, both focus only on the dynamics of the context's knowledge bases, thus not allowing the bridge rules of the contexts to change. However, as the world evolves, it is also quite natural that the way in which information flows between contexts be subject to change. For example, as bridge rules represent how contexts are accessed, and their knowledge used, changes in the level of trust of these contexts can lead to changes in the way their knowledge is used, i.e., changes in the bridge rules that appeal to those contexts. Even if not triggered by issues such as trust, we may simply want to change the bridge rules, e.g., by adding exceptions to existing ones. To address this drawback, we should allow the initial set of bridge rules to undergo change, at runtime, triggered by the observation of new bridge rules, which act as updates to the previous ones. This update naturally needs to go beyond the simple addition of the new rules since consistency between new and previously existing bridge rules needs to be ensured.

In this paper we fill this gap by presenting an extension to eMCSs, called bridge-rule evolving Multi-Context Systems (beMCSs), which combines the ability to both react to, and reason in the presence of commonly temporary dynamic observations, and evolve by incorporating new knowledge, inherited from eMCS, with the ability to update the bridge rules of each context, taking into account an incoming stream of observed bridge rules. We show that our framework satisfies several desirable properties and how the important problem of consistency management can be dealt with.

The remainder of this paper is structured as follows. After introducing the main concepts regarding mMCSs, we define beMCSs and prove some properties of the framework. Then, we discuss consistency management. We conclude with discussing related work and possible future directions.

Example 1 (Running example). Throughout this paper, we will illustrate some of our concepts using the scenario of an airport, where there is an agent responsible for its security.[1] Such an agent should build its knowledge based on existing knowledge distributed across several heterogeneous knowledge sources. First of all, the agent should have access to an airport ontology, which describes airport concepts, e.g., terminals, gates, etc., to avoid creating and maintaining its own. Another important component is that of the security norms, usually published by a national authority, that describe what is obligatory, permitted and forbidden with respect to airport security. As in any true multi-agent system, the agent should have a model of every other relevant agent, e.g., other security agents working in cooperation. Here, such models are meant to be the idealization the security agent has about the other agents based on observations about and communication with them. In this scenario, the security agent will have to react and evolve given incoming streams of information, e.g., provided by sensors (e.g., passengers arriving to the airport, images from cameras, etc.) or from other agents, but it should also be able to change its specification regarding how all this information flows between contexts and is combined.

[1] This example is partially inspired by an example presented in [25].

2 Preliminaries: Managed Multi-Context Systems

Following [9], a Multi-Context System (MCS) consists of a collection of components, each of which contains knowledge represented in some *logic*, defined as a triple $L = \langle \mathbf{KB}, \mathbf{BS}, \mathbf{ACC} \rangle$ where $\mathbf{KB}$ is the set of well-formed knowledge bases of L, $\mathbf{BS}$ is the set of possible belief sets, and $\mathbf{ACC} : \mathbf{KB} \to 2^{\mathbf{BS}}$ is a function describing the semantics of L by assigning to each knowledge base a set of acceptable belief sets. We assume that each element of $\mathbf{KB}$ and $\mathbf{BS}$ is a set, and we define $F = \{s : s \in kb \wedge kb \in \mathbf{KB}\}$.

In addition to the knowledge base in each component, *bridge rules* are used to interconnect the components, specifying what knowledge to assert in one component given certain beliefs held in the components of the MCS. Bridge rules in MCSs only allow adding information to the knowledge base of their corresponding context. In [10], an extension of MCSs, called managed Multi-Context Systems (mMCSs), is introduced in order to allow other types of operations to be performed on a knowledge base. For that purpose, each context of an mMCS is associated with a *management base*, which is a set of operations that can be applied to the possible knowledge bases of that context. Given a management base OP and a logic L, let $OF = \{op(s) : op \in OP \wedge s \in F\}$ be the *set of operational formulas* over OP and L. Each context of an mMCS gives semantics to operations in its management base using a *management function* over a logic L and a management base OP, $mng : 2^{OF} \times \mathbf{KB} \to (2^{\mathbf{KB}} \setminus \{\emptyset\})$, i.e., $mng(Op, kb)$ is the (non-empty) set of possible knowledge bases that result from applying the operations in Op to the knowledge base kb. We assume that $mng(\emptyset, kb) = \{kb\}$.

Let $L = \langle L_1, \ldots, L_n \rangle$ be a sequence of logics and OP_i a management base. We denote by OF_i the set of operational formulas over OP_i and L_i. Then a *bridge rule* σ for L_i and OP_i over L, $1 \leq i \leq n$, is a rule of the form $op(s) \leftarrow a_1, \ldots, a_k, \mathbf{not}\ a_{k+1}, \ldots, \mathbf{not}\ a_n$, where $op(s) \in OF_i$, and, for each $1 \leq i \leq n$, a_i is of the form $(r{:}b)$ where $r \in \{1, \ldots, n\}$ and b is a belief formula of L_r. Given a bridge rule σ of the above form, the head and the body of σ are defined as $H(\sigma) = op(s)$ and $B(\sigma) = \{a_1, \ldots, a_k, \mathbf{not}\ a_{k+1}, \ldots, \mathbf{not}\ a_n\}$, respectively. As we will specify below, intuitively, the operational formula in the head will be applied to the knowledge base using mng if all elements in the body are in accordance with the beliefs held in the corresponding contexts r.

Putting all the above together, a *managed Multi-Context System* (mMCS) is a sequence $M = \langle C_1, \ldots, C_n \rangle$, where each C_i, $1 \leq i \leq n$, called a *managed context*, is defined as $C_i = \langle L_i, kb_i, br_i, OP_i, mng_i \rangle$ where

- $L_i = \langle \mathbf{KB}_i, \mathbf{BS}_i, \mathbf{ACC}_i \rangle$ is a logic
- $kb_i \in \mathbf{KB}_i$
- OP_i is a management base
- br_i is a set of bridge rules for L_i and OP_i over $\langle L_1, \ldots, L_n \rangle$
- mng_i is a management function over L_i and OP_i.

For the sake of readability, we consider a slightly restricted version of mMCSs where each $\mathbf{ACC}_i$ is a function and not a set of functions as for logic suites [10].

Example 2 (Ctd.). We now briefly sketch an mMCS for the airport scenario as outlined in Sect. 1. The idea is not to present a full detailed description of the mMCS, but rather to describe parts of the example which will help us illustrating our approach. We present a simplified modeling of the airport security agent using an mMCS with five contexts, one for each relevant entity: the airport ontology, the normative entity, the security agent, and two other agents, agent A and agent B, which work in cooperation with the security agent. The airport ontology is a Description Logic (DL) [5] context, since DLs are well-suited for hierarchical information. Both the normative institution and the security agent are modeled by Logic Programming (LP) [18] contexts, since LP is well-suited to represent rule-based languages. For simplicity, we also assume that the representation of information the security agent has about the other two agents is modeled by a context in classical logic. We now present part of the configuration of the knowledge bases of the five contexts and refer for the (standard) definitions of their logics to [13] and [10]. The knowledge base of the ontology context includes taxonomic information based on usual airport vocabulary, such as *Onboard*, *Flight*, *Passenger*. The set of taxonomic axioms, usually denoted the TBox of the ontology, contains, for example, the axiom $\exists$ *Onboard*.$\top \sqsubseteq$ *Passenger*, stating that someone onboard is a passenger. Besides the hierarchical information in the TBox, with a more static nature, the ontology can also have more dynamic data, in this case about flights, airlines, etc., usually denoted the ABox of the ontology. The ABox contains, for example, *Flight(KM101)* and *Onboard(John,KM101)*.

The knowledge base of the normative context contains the LP rules:

$$\mathit{TakeOffNotAllowed(f)} \leftarrow \mathit{Flight(f)}, \mathit{IntDest(f)}, \mathit{Onboard(x,f)}, \textbf{not } \mathit{HasPassport(x)}$$
$$\mathit{HasPassport(x)} \leftarrow \mathit{Passenger(x)}, \mathit{Passport(p)}, \mathit{Carries(x,p)}$$

The first rule states that an international flight is not allowed to take off if there is someone onboard which is not known to carry a passport. The second rule defines when a passenger has a passport.

The knowledge base of the security agent includes the rule

$$\mathit{Investigate(f)} \leftarrow \mathit{BoardingProblem(f)}, \textbf{not } \mathit{UnderInvestigation(f)}$$

stating that the agent should investigate a flight for which there is a boarding problem and it is not known that the flight is already being investigated (by another agent). The knowledge bases of agents A and B contain the formula *Investigating(f)* whenever they are investigating flight f. Then, as we will see later, *UnderInvestigation(f)* will be added to the knowledge base of the security agent's context via bridge rules whenever *Investigating(f)* is believed true in the context of either agent A or agent B.

For an mMCS $M = \langle C_1, \dots, C_n \rangle$, a *belief state of* M is a sequence $S = \langle S_1, \dots, S_n \rangle$ such that each S_i is an element of $\mathbf{BS}_i$. For a bridge literal $(r:b)$, $S \models (r:b)$ if $b \in S_r$ and $S \models \textbf{not}\,(r:b)$ if $b \notin S_r$; for a set of bridge literals B, $S \models B$ if $S \models L$ for every $L \in B$. We say that a bridge rule σ of a context C_i is *applicable given a belief state* S *of* M if S satisfies $B(\sigma)$. We can then define

$app_i(S)$, the set of heads of bridge rules of C_i which are applicable in S, by setting $app_i(S) = \{H(\sigma) : \sigma \in br_i \wedge S \models B(\sigma)\}$.

Equilibria are belief states that simultaneously assign an acceptable belief set to each context in the mMCS such that the applicable operational formulas in bridge rule heads are taken into account. Let $M = \langle C_1, \ldots, C_n \rangle$ be an mMCS and $S = \langle S_1, \ldots, S_n \rangle$ a belief state of M. Then, S is an *equilibrium* of M if, for every $1 \leq i \leq n$, we have $S_i \in \mathbf{ACC}_i(kb)$ for some $kb \in mng_i(app_i(S), kb_i)$.

3 Evolving Bridge Rules

Evolving Multi-Context Systems (eMCSs) [21] admit so-called *observation contexts* whose knowledge bases are constantly changing over time according to the observations made, similar, e.g., to streams of data from sensors.[2] As outlined in Sect. 1, such eMCSs do not consider potential changes in the bridge rules that may be the result of simple observations, a learning process of the agent itself or indicated by the programmer at runtime. In this section, we introduce beMCSs, that extend eMCSs by also allowing that each context receives an incoming stream of sets of such bridge rules, which is meant to incrementally update the set of bridge rules of the context. For that purpose, rather than first recalling eMCSs and then presenting its extension beMCSs, we present the combined formalism beMCSs right away and point out concrete differences when discussing the formalization of updating bridge rules.

Following [21], regarding the observations made by the observation contexts, these will also affect the other contexts by means of the bridge rules. As we will see, such effect can either be instantaneous and temporary, i.e., limited to the current time instant, similar to (static) mMCSs, where the body of a bridge rule is evaluated in a state that already includes the effects of the operation in its head, or persistent, but only affecting the next time instant. To achieve the latter, we extend the operational language with a unary meta-operation *next* that can only be applied on top of operations.

Definition 1. *The* evolving operational language *over a management base OP and a logic L is defined as* $eOF = OF \cup \{next(op(s)) : op(s) \in OF\}$.

The idea of observation contexts is that each such context has a language describing the set of possible observations of that context, along with its current observation. The elements of the language of the observation contexts can then be used in the body of bridge rules to allow contexts to access the observations. Formally, an *observation context* is a tuple $O = \langle \Pi_O, \pi \rangle$ where Π_O is the *observation language* of O and $\pi \subseteq \Pi_O$ is its *current observation.*

We can now adapt beMCSs from eMCSs.

Definition 2. *A* beMCS *is a sequence* $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_\ell \rangle$*, such that each* $O_j = \langle \Pi_{O_j}, \pi_j \rangle$*,* $j \in \{1, \ldots, \ell\}$*, is an* observation context*, and each* evolving context C_i*,* $i \in \{1, \ldots, n\}$*, is defined as* $C_i = \langle L_i, kb_i, br_i, OP_i, mng_i \rangle$ *where*

[2] For simplicity of presentation, discrete steps in time are considered.

– *$L_i = \langle \mathbf{KB}_i, \mathbf{BS}_i, \mathbf{ACC}_i \rangle$ is a logic*
– *$kb_i \in \mathbf{KB}_i$*
– *br_i is a set of bridge rules of the form*
$$H(\sigma) \leftarrow a_1, \ldots, a_k, \mathbf{not}\ a_{k+1}, \ldots, \mathbf{not}\ a_n \tag{1}$$
such that $H(\sigma) \in eOF_i$, and each a_i, $i \in \{1, \ldots, n\}$, is either of the form $(r\!:\!b)$ with $r \in \{1, \ldots, n\}$ and b a belief formula of L_r, or of the form $(r@o)$ with $r \in \{1, \ldots, \ell\}$ and $o \in \Pi_{O_r}$
– *OP_i is a management base*
– *mng_i is a management function over L_i and OP_i.*

We denote by $\mathbf{BR}_i$ the set of possible bridge rules of the form (1) for C_i.

Let $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_\ell \rangle$ be a beMCS. As for mMCSs, the notion of *belief state for M_e* is defined as a sequence $S = \langle S_1, \ldots, S_n \rangle$ such that, for each $1 \leq i \leq n$, we have $S_i \in \mathbf{BS}_i$.

The notion $app_i(S)$ of the set of heads of bridge rules of C_i which are applicable in a belief state $S = \langle S_1, \ldots, S_n \rangle$, cannot be directly transferred from mMCSs to beMCS since bridge rule bodies can now contain atoms of the form $(r@o)$, whose satisfaction depends on the current observation.

The satisfaction of bridge literals of the form $(r\!:\!b)$ carries over from mMCSs. The satisfaction of bridge literal of the form $(r@b)$ depends on the current observations, i.e., we have that $S \models (r@o)$ if $o \in \pi_r$ and $S \models \mathbf{not}\,(r@o)$ if $o \notin \pi_r$. As before, for a set B of bridge literals, we have $S \models B$ if $S \models L$ for every $L \in B$.

We say that a bridge rule σ of a context C_i is *applicable given a belief state S for M_e* if $S \models B(\sigma)$. Then, given a belief state S for M_e and a set br of bridge rules for M_e, we can define $app(S, br) = \{H(\sigma) : \sigma \in br \text{ and } S \models B(\sigma)\}$, the set of heads of bridge rules in br which are applicable given S.

Recall that the heads of bridge rules in a beMCS are more expressive than in an mMCS, since they may be of two types: those that contain *next* and those that do not. As already mentioned, the former are to be applied to the current knowledge base and not persist, whereas the latter are to be applied in the next time instant and persist. Therefore, we distinguish these two subsets of $app(S, br)$ by setting:

Definition 3. *Let $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_\ell \rangle$ be a beMCS, br a set of bridge rules with $br \subseteq \bigcup_i \mathbf{BR}_i$, and S a belief state for M_e. Then, consider the sets:*
– *$app^{next}(S, br) = \{op(s) : next(op(s)) \in app(S, br)\}$*
– *$app^{now}(S, br) = \{op(s) : op(s) \in app(S, br)\}$*

This definition is a generalization of Def. 3 [21] to arbitrary sets of bridge rules. Nevertheless, for the set of bridge rules of a context C_i, br_i, we can use the notation $app_i^{next}(S)$ and $app_i^{now}(S)$ as in [21] to denote, respectively, $app^{next}(S, br_i)$ and $app^{now}(S, br_i)$.

Note that we can easily model a scenario where we want an effect to be instantaneous and persistent. This can be achieved using two bridge rules with identical body, one with and one without *next* in the head.

Example 3 (Ctd.). We now sketch a beMCS modeling the airport security agent. Let $M_e = \langle C_1, C_2, C_3, C_4, C_5, O_1 \rangle$ be composed of five evolving contexts C_1, C_2,

C_3, C_4 and C_5, corresponding to the airport ontology, the normative entity, the security agent, agent A, and agent B, respectively, whose knowledge bases are partially given in Example 2. The observation context, O_1, now models incoming information arriving to the system, which allows each context, through its bridge rules, to react and evolve given such observations. For simplicity, we consider just one observation context, which is responsible for monitoring flight gates, and omit here more sophisticated observations, e.g., readings of electronic passports, images from cameras, etc. The language of O_1 contains elements such as *enterPlane(John,1234)*, stating that *John* has just entered the plane with identification *1234*.

The ontology context C_1 contains the following bridge rule:

$$\textit{next(add(Onboard(x,f)))} \leftarrow 1@\textit{EnterPlane(x,p)}, 1\!:\!\textit{Assigned(p,f)}$$

stating that if it is observed that a person enters a plane which is assigned to a flight, then this person is onboard that flight. Note the use of *next* in the head of the rule to guarantee that *Onboard(x,f)* is persistently added to the ontology.

The normative context C_2 contains the following bridge rules, importing the relevant information from the ontology context C_1:

$$\textit{upd(Flight(f)}\leftarrow\textit{)} \leftarrow 1\!:\!\textit{Flight(f)}$$
$$\textit{upd(Onboard(x,f)}\leftarrow\textit{)} \leftarrow 1\!:\!\textit{Onboard(x,f)}$$

Note that, to not duplicate information already in the ontology, the above rules only import information temporarily to C_2, without using the operator *next*.

The security agent context C_3 has the following bridge rules:

$$\textit{upd(UnderInvestigation(f)}\leftarrow\textit{)} \leftarrow 4\!:\!\textit{Investigating(f)}$$
$$\textit{upd(UnderInvestigation(f)}\leftarrow\textit{)} \leftarrow 5\!:\!\textit{Investigating(f)}$$

stating that some flight is under investigation if some of the other agents is already investigating it. Note that these rules are not meant to be persistent, since whenever *Investigating(f)* does not hold for the other agents, *UnderInvestigation(f)* should immediately not hold for the security agent.

The context of agent A, C_4, has an empty set of bridge rules, and the context of agent B, C_5, contains the following bridge rules:

$$\textit{add(goHelpSA)} \leftarrow 3\!:\!\textit{NeedHelp}$$
$$\textit{add(goHelpA)} \leftarrow 4\!:\!\textit{NeedHelp}$$

stating that agent B should help any of the other agents that asked for help.

Similar to equilibria in mMCS, the (static) equilibrium is defined to incorporate instantaneous effects based on $app_i^{now}(S)$ alone.

Definition 4. *Let $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_\ell \rangle$ be a beMCS. A belief state $S = \langle S_1, \ldots, S_n \rangle$ for M_e is a static* equilibrium *of M_e iff for each $1 \leq i \leq n$, there exists some $kb \in mng_i(app_i^{now}(S), kb_i)$ such that $S_i \in \mathbf{ACC}_i(kb)$.*

To assign meaning to a beMCS evolving over time we consider sequences of belief states, evolving belief states, each referring to a subsequent time instant.

Definition 5. *Let M_e be a beMCS. An* evolving belief state *of size s for M_e is a sequence $\mathcal{S} = \langle S^1, \ldots, S^s \rangle$ where each S^j, $1 \leq j \leq s$, is a belief state for M_e.*

So far, apart from the generalization in Def. 3, the notions for eMCSs and beMCSs coincide. Next, we discuss how to update a beMCS, which unlike for eMCSs requires considering how to update bridge rules.

To be able to update the knowledge bases and the sets of bridge rules of the evolving contexts, we need the following notation. Given an evolving context C_i, a knowledge base $k \in \mathbf{KB}_i$ and a set of bridge rules $b \subseteq \mathbf{BR}_i$, we denote by $C_i[k, b]$ the evolving context in which kb_i and br_i are replaced by k and b respectively, i.e., $C_i[k, b] = \langle L_i, k, b, OP_i, mng_i \rangle$. For an observation context O_i, given a set $\pi \subseteq \Pi_{O_i}$ of observations for O_i, we denote by $O_i[\pi]$ the observation context in which its current observation is replaced by π, i.e., $O_i[\pi] = \langle \Pi_{O_i}, \pi \rangle$.

To enable beMCSs to react to incoming observations and evolve, an observation sequence defined in the following has to be processed. The idea is that, at each time instant, we have two types of observations. On the one hand, we have a set of observations for each observation context O_i, which is meant to replace its current observation. On the other hand, we have a set of bridge rules for each evolving context C_i, which is meant to update the set of bridge rules br_i of C_i.

Recall from the Introduction that there are two main motivations for updating the set of bridge rules of a context. One the one hand, we may want to substitute an existing rule with a more recent one, since, based for example on a change of trust, we may want to change the sources of information in a rule. On the other hand, we may want to add exceptions to existing rules. Given such motivations, and since the bridge rules in a beMCS, as in the case of mMCSs, are similar to logic programming rules, we build the updates of bridge rules upon the work done in updates of logic programs, namely on Dynamic Logic Programs (DLP) [3]. In this approach, the use of default negation in the head of rules is fundamental to allow explicit rejection of rules and also the introduction of exceptions to existing rules. Therefore, to update the set of bridge rules of a context C_i we consider more expressive bridge rules, which allow default negation in the head. Formally, for each $1 \leq i \leq n$, we consider $e\mathbf{BR}_i$, the set of *evolving bridge rules for* C_i, defined as $e\mathbf{BR}_i = \mathbf{BR}_i \cup \{\mathbf{not}\ H(\sigma) \leftarrow B(\sigma) : \sigma \in \mathbf{BR}_i\}$.

We can now define the notion of observation sequence for a beMCS.

Definition 6. *Let $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_\ell \rangle$ be a beMCS. An* observation sequence *for M_e is a sequence $Obs = \langle \mathcal{O}^1, \ldots, \mathcal{O}^m \rangle$, where, for each $1 \leq j \leq m$, $\mathcal{O}^j = \langle o^j, obr^j \rangle$, is an* instant observation *containing* observations $o^j = \langle o^j_1, \ldots, o^j_\ell \rangle$ *such that, for each $1 \leq i \leq \ell$, $o^j_i \subseteq \Pi_{O_i}$, and* observed bridge rules *$obr^j = \langle obr^j_1, \ldots, obr^j_n \rangle$ such that, for each $1 \leq i \leq n$, $obr^j_i \subseteq e\mathbf{BR}_i$.*

Our aim now is to show how, given an observation sequence $\mathcal{O}$, the beMCS M_e is able to react and evolve. As mentioned before, the observation contexts evolve by replacing their set of current observations according to the observation sequence. We still need to define how the bridge rules of each evolving context are updated given an observation sequence. Our goal is to define for each context C_i the set $\mathit{Upd}(S, br, B)$ of possible updates of a set $br \subseteq \mathbf{BR}_i$ of bridge rules by a sequence $B = \langle \mathit{obr}_i^1, \ldots, \mathit{obr}_i^k \rangle$ of sets of evolving bridge rules with $\mathit{obr}_i^j \subseteq e\mathbf{BR}_i$ for $1 \leq j \leq k$, given the belief state S for M_e. The idea is to combine two update mechanisms, both building upon the notion of rejected bridge rule: one is based on the existence of an explicit conflict between an operation and its default negation in the head of bridge rules; the other is based on an implicit notion of inconsistency between operational formulas, which depends on each context.

To define update operators based on conflicts which arise due to the use of default negation in the head of rules we follow the ideas of DLP [3]. The intuition is that a rule σ is rejected in state S if there is a more recent rule which is applicable in S, and whose head is the default negation of the head of σ. Formally, let M_e be a beMCS, S a belief state for M_e, and $B = \langle \mathit{obr}_i^1, \ldots, \mathit{obr}_i^k \rangle$ a sequence of sets of evolving bridge rules of some evolving context C_i of M_e, i.e., each $\mathit{obr}_i^j \subseteq e\mathbf{BR}_i$. We can then define $\mathit{ExpRej}(S, \langle \mathit{obr}_i^1, \ldots, \mathit{obr}_i^k \rangle)$, the sequence of sets of bridge rules that result from B by removing the rules explicitly rejected by a more recent rule, by setting $\mathit{ExpRej}(S, \langle \mathit{obr}_i^1, \ldots, \mathit{obr}_i^k \rangle) = \langle Br^1, \ldots, Br^k \rangle$ such that, for each $1 \leq j \leq k$, we have $Br^j = (\mathit{obr}_i^j \cap \mathbf{BR}_i) \setminus \mathit{Rej}^j$ where

$$\begin{aligned} \mathit{Rej}^j = \{\sigma \in \mathit{obr}_i^j : \text{there is } \sigma' \in \mathit{obr}_i^{j'} \text{ with } j' > j \text{ such that} \\ H(\sigma') = \mathbf{not}\ H(\sigma) \text{ and } S \models B(\sigma')\}. \end{aligned}$$

Example 4 (Ctd.). Continuing the airport example, suppose that the security agent no longer trusts agent A. In that case he wants to cancel the existing rule, and still investigate a flight even though agent A is already investigating it. In that case, he can update its set of bridge rules with the evolving bridge rule:

$$\mathbf{not}\ \mathit{upd(UnderInvestigation(f)} \leftarrow) \leftarrow 4{:}\mathit{Investigating(f)}$$

In this case, whenever *Investigating(f)* is true in C_4 this more recent bridge rule rejects the initial one, and therefore the security agent does not update its knowledge base with the LP fact *UnderInvestigation(f)*$\leftarrow$.

We now focus on the notion of update of bridge rules based on a notion of implicit inconsistency between operational formulas. This makes sense since the language of the heads of bridge rules is so general and potentially quite expressive. Therefore, besides the explicit notion of rejected bridge rule mentioned above, we also consider a notion of rejected bridge rule based on a notion of inconsistency over operational formulas, which depends on each evolving context. More precisely, we assume that, for each evolving context C_i of M_e, there is a relation $\mathit{Inc}_i : \mathbf{BS}_i \times 2^{OF_i}$. The intuitive idea is that $\langle S_i, \mathit{Op} \rangle \in \mathit{Inc}_i$ if Op is an inconsistent set of operational formulas w.r.t. S_i. This notion of inconsistent set of operations depends on each context. Interesting examples include the case in which there is a conflict between two contrary operations, for example adding and removing, $\mathit{add}(p)$ and $\mathit{rm}(p)$. We can also have conflicts with the same operation, for example addition of two complementary literals, $\mathit{add}(p)$ and $\mathit{add}(\neg p)$.

Just as a last example, let C_i be a Classical Logic (CL) context and suppose that $OP_i = \{add\}$ where add is simple addition. We could then define Inc_i based on whether, for a set of operational formulas Op, the set $\{\varphi : add(\varphi) \in Op\}$ is consistent in CL. Given such definition, we have, for every S_i, for example that $\langle S_i, \{add(a \Rightarrow b), add(a), add(\neg b)\}\rangle \in Inc_i$.

Note that, contrarily to the above examples, there are cases where conflicts depend on the belief state. Take, for example, the case of Logic Programming (LP). The notion of conflict between LP rules depends on the belief state.

We assume that the notion of operational inconsistency satisfies the following natural condition. Given a belief state S, for all $1 \leq i \leq n$, we assume that the set $Inc_i^S = \{Op : \langle S_i, Op\rangle \in Inc_i\}$ is an upper set of the partially ordered set $\langle 2^{OF_i}, \subseteq\rangle$, i.e., if $\langle S_i, Op\rangle \in Inc_i$ and $Op \subseteq Op'$, then $\langle S_i, Op'\rangle \in Inc_i$.

Let br_1 and br_2 be two sets of bridge rules of a context C_i. Our aim is to define the possible sets of bridge rules that result from updating br_1 with br_2. For that, as we said, we define a notion of rejected rule based on operational inconsistency. Given a belief state S of M_e, we define $Rej(S, br_1, br_2)$, the set of sets of rejected bridge rules, as:

$$Rej(S, br_1, br_2) = \{b \subseteq br_1 : app^{now}(S, b \cup br_2) \in Inc_i^S \text{ or } \\ app^{next}(S, b \cup br_2) \in Inc_i^S\}.$$

When updating a set of rules br_1 by a set of rules br_2 we are interested in minimizing the set of rejected rules of br_1. Therefore, we consider the set $MinRej(S, br_1, br_2)$ of all minimal elements of $Rej(S, br_1, br_2)$.

Using the set of minimal set of rejected bridge rules, we can define, for a belief state S of M_e, the set of sets of acceptable bridge rules given S as:

$$Acpt(S, br_1, br_2) = \{b \subseteq br_1 : b' \not\subseteq b \text{ s.t. } \forall b' \in MinRej(S, br_1, br_2)\}.$$

The set $Acpt(S, br_1, br_2)$ is the set of all subsets of br_1 which can be consistently added to br_2 in the context of state S. As usual, when updating a set of rules br_1 by a set of rules br_2 we are interested in maximizing the set of elements of br_1 in the final result. Therefore, we define $MaxAcpt(S, br_1, br_2)$ as the set of maximal elements of $Acpt(S, br_1, br_2)$, i.e., those $b \in Acpt(S, br_1, br_2)$ for which there is no $b' \in Acpt(S, br_1, br_2)$ such that $b \subset b'$.

Example 5 (Ctd.). Recall that the context of agent B, C_5, has two bridge rules, denoted here by σ_1 and σ_2, which are meant to react to the fact that the other agents asked for help, and let $br_1 = \{\sigma_1, \sigma_2\}$. Now imagine that, for some reason (efficiency, design decision, etc.), agent B cannot help both agents at the same time. For incorporating this information, he can consider an update of br_1 by the set $br_2 = \{add(\neg(goHelpSA \wedge goHelpA)) \leftarrow 3 : NeedHelp, 4 : NeedHelp\}$. Suppose also that C_5 has the following natural inconsistency relation: for every belief state S, $\langle S_5, Op\rangle \in Inc_5$ if the set $\{p : add(p) \in Op\}$ is inconsistent in CL. Then, taking $Op = \{add(goHelpSA), add(goHelpA), add(\neg(goHelpSA \wedge goHelpA))\}$ we have, for every S, that $\langle S_5, Op\rangle \in Inc_5$. We can easily check that $MinRej(S, br_1, br_2) = \{br_1\}$. This implies that $MaxAcpt(S, br_1, br_2) =$

$\{\{\sigma_1\}, \{\sigma_2\}\}$, meaning that the possible updates of br_1 by br_2 should contain σ_1 or σ_2, but not both.

We now extend the notion of *MaxAcpt* to sequences of sets of bridge rules:

$$\begin{aligned} \mathit{MaxAcpt}(S, \langle br^1, \ldots, br^k\rangle) = \{ br_k :\ & \text{there exists } \langle b^1, \ldots, b^k\rangle \text{ satisfying} \\ & - b^1 = br^1 \\ & - b^{j+1} = br^{j+1} \cup b \text{ where } b \in \mathit{MaxAcpt}(S, br^j, br^{j+1})\}. \end{aligned}$$

The following result states the connection between the set of all maximal sets of accepted bridge rules and the set of all minimal sets of rejected rules.

Proposition 1. *Let $b \subseteq br_1$ such that $b \in \mathit{MaxAcpt}(S, br_1, br_2)$. Then we have that $br_1 \setminus (\bigcup \mathit{MinRej}(S, br_1, br_2)) \subseteq b$.*

We can now define $\mathit{Upd}(S, br, \langle br^1, \ldots, br^k\rangle)$ the set of possible results of updating the set br of bridge rules by the sequence $S = \langle br^1, \ldots, br^k\rangle$ of sets of evolving bridge rules. The idea is to combine the two update mechanisms described above: rejection based on conflict between an operation and its default negation, and rejection based on the operational inconsistency relation Inc_i.

Formally, given a set br of bridge rules of C_i, a sequence $B = \langle br^1, \ldots, br^k\rangle$ of sets of evolving bridge rules of C_i, and S a belief state of M_e, we can define the set $\mathit{Upd}(S, br, B)$ of possible results of updating br by the sequence B as:

$$\mathit{Upd}(S, br, B) = \mathit{MaxAcpt}(S, \mathit{ExpRej}(S, \langle br, br^1, \ldots, br^k\rangle)).$$

One basic property that we need to guarantee is that, when updating by a sequence of sets of bridge rules, the most recent bridge rules are always contained in every possible result of the update. The following result states this property, taking into account that bridge rules with default negation in the head cannot appear in the result of an update.

Proposition 2. *Let S be a belief state of M_e, br a set of bridge rules of C_i, and $B = \langle br_i^1, \ldots, br_i^k\rangle$ a sequence of sets of evolving bridge rules of C_i. Then, for every $b \in \mathit{Upd}(S, br, B)$, we have that $(br_i^k \cap \mathbf{BR}_i) \subseteq b$.*

Now that we have defined how the observation contexts and the sets of bridge rules evolve, we can define the notion of evolving equilibrium of a beMCS $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_\ell\rangle$ given an observation sequence $\mathit{Obs} = \langle \mathcal{O}^1, \ldots, \mathcal{O}^m\rangle$ for M_e. The intuitive idea is that, given an evolving belief state $\mathcal{S} = \langle S^1, \ldots, S^s\rangle$ for M_e, in order to check if $\mathcal{S}$ is an evolving equilibrium, we need to consider a sequence of beMCSs, $M^1, \ldots, M^s$, representing a possible evolution of M_e according to the observations in *Obs*, such that S^j is a static equilibrium of M^j. For each M^j the sets of current observations of the observation contexts are exactly their corresponding elements π_i^j in $\mathcal{O}^j$. For each of the evolving contexts C_i, its knowledge base in M^j is obtained from the one in M^{j-1} by applying the operations in $app_i^{next}(S^{j-1})$. Moreover the set of bridge rules of each evolving context is updated using $\mathit{Upd}(S^j, br_i, \langle obr_i^1, \ldots, obr_i^j\rangle)$.

Definition 7. *Let $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_\ell \rangle$ be a beMCS, $\mathcal{S} = \langle S^1, \ldots, S^s \rangle$ an evolving belief state of size s for M_e, and $Obs = \langle \mathcal{O}^1, \ldots, \mathcal{O}^m \rangle$ an observation sequence for M_e such that $m \geq s$. Then, $\mathcal{S}$ is an* evolving equilibrium *of size s of M_e given Obs iff, for each $1 \leq j \leq s$, S^j is a static equilibrium of $M^j = \langle C_1[k_1^j, b_1^j], \ldots, C_n[k_n^j, b_n^j], O_1[o_1^j], \ldots, O_\ell[o_\ell^j] \rangle$ where, for each $1 \leq i \leq n$,*

- $b_i^j \in Upd(S^j, br_i, \langle obr_i^1, \ldots, obr_i^j \rangle)$

and k_i^j is defined inductively as follows:

- $k_i^1 = kb_i$
- $k_i^{j+1} \in mng_i(app^{next}(S^j, b_i^j), k_i^j)$.

Note that the set of bridge rules of a context can change from one time instant to the other even if no bridge rule of that context is observed. This happens because the set of updates depends also on the current state. For a simple example, let $\sigma_1 = (add(p) \leftarrow)$ be a bridge rule of a context in a beMCS, and suppose that at time 1 the rule $\sigma_2 = (\textbf{not}\ add(p) \leftarrow 1{:}\textbf{not}\ q)$ is observed, which can be seen as an exception for the original rule. Suppose that q is not true in context C_1 at time 1. Then, both rules are applicable and their heads are conflicting, leading to the rejection of the first rule. Suppose that in the next time instant no bridge rule is observed, but q is true. Then, since the second rule is not applicable, the rules are not conflicting. Therefore, σ_1 is not rejected and it is now part of the set of bridge rules of C_1.

We now prove some properties of the notion of evolving equilibrium. In Def. 7, the size of the observation sequence is assumed to be greater or equal than the size of the evolving belief state. The intuition is that an equilibrium may also be defined for only a part of the observation sequence. As a consequence, any subsequence of an evolving equilibrium is still an evolving equilibrium.

Proposition 3. *Let M_e be a beMCS and $Obs = \langle \mathcal{O}^1, \ldots, \mathcal{O}^m \rangle$ an observation sequence for M_e. If $\mathcal{S} = \langle S^1, \ldots, S^s \rangle$ is an evolving equilibrium of size s of M_e given Obs, then, for each $1 \leq j \leq s$, and every $j \leq k \leq m$, $\langle S^1, \ldots, S^j \rangle$ is an evolving equilibrium of size j of M_e given the observation sequence $\langle \mathcal{O}^1, \ldots, \mathcal{O}^k \rangle$.*

It is not hard to see that an mMCS is a particular case of a beMCS with no observation context, the heads of bridge rules do not contain the operator *next*, and there are no updates to the bridge rules.

Proposition 4. *Let $M = \langle C_1, \ldots, C_n \rangle$ be an mMCS. Then, $S = \langle S_1, \ldots, S_n \rangle$ is an equilibrium of M iff $\mathcal{S} = \langle S \rangle$ is an evolving equilibrium of size 1 of M for some observation sequence $Obs = \langle \mathcal{O}^1, \ldots, \mathcal{O}^m \rangle$ for M with $m \geq 1$ and such that, for every $1 \leq i \leq n$, we have that $obr_i^1 = \emptyset$.*

4 Inconsistency Management

Inconsistency management is an important topic for frameworks that aim at integrating knowledge from different sources and has been extensively studied for MCSs and mMCSs [13,10]. In [21], inconsistency management for eMCSs is investigated, and it is shown that essential notions and results carry over from static mMCSs to dynamic eMCSs. In this section, we adapt these results from eMCSs to beMCSs and can confirm that the same favorable characteristics hold.

For the case of mMCSs, three forms of inconsistency are considered: *nonexistence of equilibria*, *local inconsistency*, and *operator inconsistency* [10]. The first form has been extensively studied for MCSs [13] and is also termed global inconsistency, while the second one deals with inconsistent belief sets potentially occurring in an equilibrium provided the contexts in the considered mMCS admit such a notion. The third form aims at detecting conflicts between operations in the heads of bridge rules.

We start by introducing the notion of (global) consistency.

Definition 8. *Let M_e be a beMCS and $Obs = \langle \mathcal{O}^1, \ldots, \mathcal{O}^m \rangle$ an observation sequence for M_e. Then, M_e is* consistent *with respect to Obs if it has an evolving equilibrium of size m given Obs.*

From Prop. 3, we immediately obtain that if there is a subsequence of *Obs* such that the considered beMCS is inconsistent, then the beMCS is also inconsistent for the entire sequence (and vice-versa).

Corollary 1. *Let M_e be a beMCS and let $Obs = \langle \mathcal{O}^1, \ldots, \mathcal{O}^m \rangle$ be an observation sequence for M_e. Then, M_e is consistent w.r.t. Obs iff M_e is consistent w.r.t. $\langle \mathcal{O}^1, \ldots, \mathcal{O}^j \rangle$ for every $1 \leq j \leq m$.*

We now focus on two notions that together are sufficient to ensure consistency. The first one focuses on the existence of an acceptable belief set for each knowledge base. Formally, an evolving context C_i in a beMCS M_e is *totally coherent* iff, for every $kb \in \mathbf{KB}_i$, $\mathbf{ACC}_i(kb) \neq \emptyset$. The second notion focuses on cycles between bridge rules. Let $B = \langle b_1, \ldots, b_n \rangle$ a tuple of sets of evolving bridge rules, one for each evolving context C_i of M_e, i.e., each $b_i \subseteq e\mathbf{BR}_i$. The idea is to describe cycles between the bridge rules that essentially may cause inconsistency. Formally we write $ref_r(i,j)$ iff r is a bridge rule of b_i and $(j:b)$ occurs in the body of r. Let $r_1, \ldots, r_k \in \bigcup_{1 \leq i \leq n} b_i$, then we say that $(r_1, \ldots, r_k)$ forms a cycle iff $ref_{r_1}(i_1, i_2), \ldots, ref_{r_{k-1}}(i_{k-1}, i_k)$, and $ref_{r_k}(i_k, i_1)$ hold. Then $B = \langle b_1, \ldots, b_n \rangle$ is *acyclic* if no such cycles exist. We can show the following.

Proposition 5. *Let $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_\ell \rangle$ be a beMCS and $Obs = \langle \mathcal{O}^1, \ldots, \mathcal{O}^m \rangle$ an observation sequence for M_e. Then, if $B = \langle b_1, \ldots, b_n \rangle$ is acyclic, where, for each $1 \leq i \leq n$, $b_i = br_i \cup (\bigcup_{j \leq m} obr_i^j)$, then M_e is consistent with respect to Obs.*

A similar property holds for mMCSs, indicating that the extension to beMCSs as such does not decrease the likelihood of existence of evolving equilibria.

An adequate treatment of local inconsistency was one of the motivations for the introduction of mMCSs, and this is also the case in beMCSs with incoming observations that also should be subject to consistency management. As described in [10], we need to assume that each context has a notion of *inconsistent* *belief state*, which usually exists or is easily definable. Assuming such notion, a knowledge base $kb_i \in \mathbf{KB}_i$ of a context C_i is said to be *consistent* if $\mathbf{ACC}_i(kb_i)$ does not contain an inconsistent belief set. A management function mng_i of a context C_i is said to be *locally consistency preserving (lc-preserving)*, if for every set $Op_i \subseteq OF_i$ and consistent knowledge base $kb_i \in \mathbf{KB}_i$, we have that every element of $mng_i(Op_i, kb_i)$ is a consistent knowledge base.

Definition 9. *Let M_e be a beMCS and $Obs = \langle \mathcal{O}^1, \ldots, \mathcal{O}^m \rangle$ an observation sequence for M_e. Then, M_e is said to be* locally consistent *with respect to Obs if every evolving equilibrium $S = \langle S^1, \ldots, S^s \rangle$ of M_e with respect to Obs is such that, for each $1 \leq j \leq s$, all belief sets in S^j are consistent.*

Recall that observations are subject to consistency management in each context. If the management functions are lc-preserving, then consistent observations do not make a consistent beMCS inconsistent.

Proposition 6. *Let $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_\ell \rangle$ be a beMCS such that, for each C_i, kb_i is consistent and mng_i is lc-preserving. Then, for every observation sequence Obs for M_e, we have that M_e is locally consistent with respect to Obs.*

Since we are assuming the existence of a notion of inconsistent set of operators for each context of a beMCS, we end this section by briefly studying this form of inconsistency. We extend to sets of bridge rules the notion of inconsistent set of operational formulas, as introduced in Sect. 3.

Definition 10. *Let $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_\ell \rangle$ be a beMCS, and S a belief state for M_e. A set $br \subseteq e\mathbf{BR}_i$ of evolving bridge rules of C_i is* inconsistent with respect to S *if $app^{now}(S, br) \in Inc_i^S$ or $app^{next}(S, br) \in Inc_i^S$. A set of bridge rules is consistent with respect to S if it is not inconsistent with respect to S.*

The following proposition guarantees the desirable property that a set of bridge rules resulting from an update with respect to a sequence of consistent sets of bridge rules is itself consistent.

Proposition 7. *Let $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_\ell \rangle$ be a beMCS, and $Obs = \langle \mathcal{O}^1, \ldots, \mathcal{O}^m \rangle$ an observation sequence for M_e, and $\mathcal{S} = \langle S^1, \ldots, S^s \rangle$ an evolving equilibrium of M_e given Obs. If each set of bridge rules in Obs is consistent with respect to S^s then, for each $1 \leq i \leq n$, we have that every element of $Upd(S^s, br_i, \langle br_i^1, \ldots, br_i^s \rangle)$ is consistent with respect to S^s.*

5 Related and Future Work

In this paper we introduced beMCS, and extension of evolving Multi-Context Systems (eMCS) [21] to allow not only the evolution of the knowledge bases of the contexts, but also of their sets of bridge rules. Closely related to eMCSs is the framework of reactive Multi-Context Systems (rMCSs) [8,15,11] inasmuch as both aim at extending mMCSs to cope with dynamic observations. The main difference between and eMCSs and rMCSs is that eMCSs have the meta operator *next* that allows for a clear separation between persistent and non-persistent effects, and also the specification of transitions based on the current state.

Another framework closely related to beMCSs is that of evolving logic programs EVOLP [2] which deals with updates of generalized logic programs, and the two frameworks of reactive ASP, one implemented as a solver *oclingo* [17] and one described in [8]. Whereas EVOLP employs an update predicate that is similar in spirit to the *next* predicate of our beMCSs, and uses the update semantics of Dynamic Logic Programming in a way that is similar to how conflicts between bridge rules dealt with in beMCSs, it does not deal with distributed heterogeneous knowledge, neither do both versions of Reactive ASP.

Regarding future work, an important non-trivial topic is the study of the notion of minimal change within an evolving equilibrium. Whereas minimal change may be desirable to obtain more coherent evolving equilibria, there are also arguments against adopting a one-size-fits-all approach embedded in the semantics. Different contexts, i.e., KR formalisms, may require different notions of minimal change, or even require to avoid it – e.g., suppose we want to represent some variable that can non-deterministically take one of two values at each time instant: minimal change could force a constant value.

Also interesting is to study how to perform AGM style belief revision at the (semantic) level of the equilibria, as in Wang et al [34], though different since knowledge is not incorporated in the contexts.

Another important issue open for future work is a more fine-grained characterization of updating bridge rules (and knowledge bases) in light of the encountered difficulties when updating rules [30,31,33] and the combination of updates over various formalisms [31,32].

Finally, we may also consider the generalization of the notions of minimal and grounded equilibria [9] to beMCSs to avoid, e.g., self-supporting cycles introduced by bridge rules, or the use of preferences to deal with several evolving equilibria a beMCS can have for the same observation sequence.

Acknowledgments. We would like to thank the referees for their comments, which helped improve this paper considerably. Matthias Knorr and João Leite were partially supported by FCT under project "ERRO – Efficient Reasoning with Rules and Ontologies" (PTDC/EIA-CCO/121823/2010). Ricardo Gonçalves was supported by FCT grant SFRH/BPD/47245/2008 and Matthias Knorr was also partially supported by FCT grant SFRH/BPD/86970/2012.

References

1. Alberti, M., Gomes, A.S., Gonçalves, R., Leite, J., Slota, M.: Normative systems represented as hybrid knowledge bases. In: Leite, J., Torroni, P., Ågotnes, T., Boella, G., van der Torre, L. (eds.) CLIMA XII 2011. LNCS, vol. 6814, pp. 330–346. Springer, Heidelberg (2011)

2. Alferes, J.J., Brogi, A., Leite, J., Moniz Pereira, L.: Evolving logic programs. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 50–61. Springer, Heidelberg (2002)
3. Alferes, J., Leite, J., Pereira, L., Przymusinska, H., Przymusinski, T.: Dynamic logic programming. In: Cohn, A., Schubert, L., Shapiro, S. (eds.) KR, pp. 98–111. Morgan Kaufmann (1998)
4. Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N.: Stream reasoning and complex event processing in ETALIS. Semantic Web 3(4), 397–407 (2012)
5. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
6. Barbieri, D., Braga, D., Ceri, S., Valle, E., Grossniklaus, M.: C-SPARQL: a continuous query language for RDF data streams. Int. J. Semantic Computing 4(1), 3–25 (2010)
7. Benerecetti, M., Giunchiglia, F., Serafini, L.: Model checking multiagent systems. J. Log. Comput. 8(3), 401–423 (1998)
8. Brewka, G.: Towards reactive multi-context systems. In: Cabalar, P., Son, T.C. (eds.) LPNMR 2013. LNCS, vol. 8148, pp. 1–10. Springer, Heidelberg (2013)
9. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: AAAI, pp. 385–390. AAAI Press (2007)
10. Brewka, G., Eiter, T., Fink, M., Weinzierl, A.: Managed multi-context systems. In: Walsh, T. (ed.) IJCAI, pp. 786–791. IJCAI/AAAI (2011)
11. Brewka, G., Ellmauthaler, S., Pührer, J.: Multi-context systems for reactive reasoning in dynamic environments. In: Schaub, T., Friedrich, G., O'Sullivan, B. (eds.) ECAI. IOS Press (to appear, 2014)
12. Dragoni, A., Giorgini, P., Serafini, L.: Mental states recognition from communication. J. Log. Comput. 12(1), 119–136 (2002)
13. Eiter, T., Fink, M., Schüller, P., Weinzierl, A.: Finding explanations of inconsistency in multi-context systems. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) KR. AAAI Press (2010)
14. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. Artif. Intell. 172(12-13), 1495–1539 (2008)
15. Ellmauthaler, S.: Generalizing multi-context systems for reactive stream reasoning applications. In: Jones, A.V., Ng, N. (eds.) ICCSW. OASICS, vol. 35, pp. 19–26. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2013)
16. Gebser, M., Grote, T., Kaminski, R., Obermeier, P., Sabuncu, O., Schaub, T.: Stream reasoning with answer set programming: Preliminary report. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) KR. AAAI Press (2012)
17. Gebser, M., Grote, T., Kaminski, R., Schaub, T.: Reactive answer set programming. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 54–66. Springer, Heidelberg (2011)
18. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Gen. Comput. 9(3/4), 365–386 (1991)
19. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics or: How we can do without modal logics. Artif. Intell. 65(1), 29–70 (1994)
20. Gonçalves, R., Alferes, J.J.: Parametrized logic programming. In: Janhunen, T., Niemelä, I. (eds.) JELIA 2010. LNCS, vol. 6341, pp. 182–194. Springer, Heidelberg (2010)
21. Gonçalves, R., Knorr, M., Leite, J.: Evolving multi-context systems. In: Schaub, T., Friedrich, G., O'Sullivan, B. (eds.) ECAI. IOS Press (to appear, 2014)

22. Homola, M., Knorr, M., Leite, J., Slota, M.: MKNF knowledge bases in multi-context systems. In: Fisher, M., van der Torre, L., Dastani, M., Governatori, G. (eds.) CLIMA XIII 2012. LNCS, vol. 7486, pp. 146–162. Springer, Heidelberg (2012)
23. Ivanov, V., Knorr, M., Leite, J.: A query tool for $\mathcal{EL}$ with non-monotonic rules. In: Alani, H., et al. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 216–231. Springer, Heidelberg (2013)
24. Knorr, M., Alferes, J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artif. Intell. 175(9-10), 1528–1554 (2011)
25. Knorr, M., Slota, M., Leite, J., Homola, M.: What if no hybrid reasoner is available? Hybrid MKNF in multi-context systems. J. Log. Comput. (2013)
26. Lécué, F., Pan, J.: Predicting knowledge in an ontology stream. In: Rossi, F. (ed.) IJCAI. IJCAI/AAAI (2013)
27. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5) (2010)
28. Roelofsen, F., Serafini, L.: Minimal and absent information in contexts. In: Kaelbling, L., Saffiotti, A. (eds.) IJCAI, pp. 558–563. Professional Book Center (2005)
29. Sabater, J., Sierra, C., Parsons, S., Jennings, N.R.: Engineering executable agents using multi-context systems. J. Log. Comput. 12(3), 413–442 (2002)
30. Slota, M., Leite, J.: On semantic update operators for answer-set programs. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) ECAI. Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 957–962. IOS Press (2010)
31. Slota, M., Leite, J.: Robust equivalence models for semantic updates of answer-set programs. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) KR. AAAI Press (2012)
32. Slota, M., Leite, J.: A unifying perspective on knowledge updates. In: del Cerro, L.F., Herzig, A., Mengin, J. (eds.) JELIA 2012. LNCS, vol. 7519, pp. 372–384. Springer, Heidelberg (2012)
33. Slota, M., Leite, J.: The rise and fall of semantic rule updates based on SE-models. TPLP (to appear, 2014)
34. Wang, Y., Zhuang, Z., Wang, K.: Belief change in nonmonotonic multi-context systems. In: Cabalar, P., Son, T.C. (eds.) LPNMR 2013. LNCS, vol. 8148, pp. 543–555. Springer, Heidelberg (2013)