Evolving Multi-Context Systems

Ricardo Gonçalves and Matthias Knorr and João Leite

Abstract. Managed Multi-Context Systems (mMCSs) provide a general framework for integrating knowledge represented in heterogeneous KR formalisms. However, mMCSs are essentially static as they were not designed to run in a dynamic scenario. In this paper, we introduce evolving Multi-Context Systems (eMCSs), a general and flexible framework which inherits from mMCSs the ability to integrate knowledge represented in heterogeneous KR formalisms, and at the same time is able to both react to, and reason in the presence of commonly temporary dynamic observations, and evolve by incorporating new knowledge. We show that eMCSs are indeed very general and expressive enough to capture several existing KR approaches that model dynamics of knowledge.

1 Introduction

Multi-Context Systems (MCSs) were introduced in [7], building on the work in [15, 19], to address the need for a general framework that integrates knowledge bases expressed in heterogeneous KR formalisms. Intuitively, instead of designing a unifying language to which other languages could be translated, in an MCS the different formalisms and knowledge bases are considered as modules, and means are provided to model the flow of information between them.

More specifically, an MCS consists of a set of contexts, each of which is a knowledge base in some KR formalism, such that each context can access information from the other contexts using so-called bridge rules. Such non-monotonic bridge rules add their heads to the context’s knowledge base provided the queries (to other contexts) in their bodies are successful. Managed Multi-Context Systems (mMCSs) were introduced in [8] to provide an extension of MCSs by allowing operations, other than simple addition, to be expressed in the heads of bridge rules. This allows mMCSs to properly deal with the problem of consistency management within contexts.

One recent challenge for KR languages is to shift from static application 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 [13, 12], C-SPARQL [5], Ontology Streams [18] and ETALIS [3], 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. With mMCSs, it is already possible to reason with such information – it can simply be treated as belonging to some new (observation) contexts which, together with some additional bridge rules, could influence what currently follows from the mMCS.

However, the requirements of such dynamic systems are more substantial. Surely, we want these observations to influence the current semantics of the mMCS, but, at the same time, to somehow be able to change the contexts of the mMCS in a more permanent way, making them evolve into a new (updated) version. This could simply be achieved by adding all these observations to some of the contexts, but doing so is often not desirable – for example, the stream of raw data may contain many facts that are irrelevant to some database (context) in the mMCS. Perhaps more importantly, simply adding even only some of these observations to a context might not be adequate, requiring more sophisticated operations such as belief revision or update – for example, some new observed (more reliable) fact may conflict with an old (less reliable) fact stored in some context, in which case we need to both add the new one and delete the old one; or some newly observed exception to a rule may require a change in the preconditions of that rule. Additionally, such transitions where contexts evolve from one state to the next should be able to depend on the earlier state of the system – for example, the observation that a light switch was flipped should result in a transition from a state (encoded in some context) where the light is on if it was previously off, and vice versa. Finally, the KR formalism in which observations are encoded may be different from the one of the context in which they need to be incorporated, thus requiring some form of conversion.

From these requirements, it is clear that we need to distinguish two different ways in which contexts needs to react to incoming observations. On the one hand, they need to react by allowing observations to influence the current state, and on the other hand, they need to react by adopting some more enduring changes that persist beyond the temporal scope of the observations, and both should be subject to consistency management.

With these requirements in mind, in this work, we propose a dynamic extension of mMCSs, called evolving Multi-Context Systems (eMCSs), a general and flexible framework which inherits from mMCSs the ability to integrate and manage knowledge represented in heterogeneous KR formalisms, and adds to it the possibility to incorporate, with different levels of persistence and through different belief change operations, knowledge obtained from dynamic observations. Just like an mMCS, an eMCS is composed of a collection of components, each of which contains knowledge represented in some logic, interconnected by bridge rules which can specify different ways to share knowledge. Some contexts of an eMCS called observation contexts, are reserved for dynamic incoming observations, changing at each state according to what is observed. More importantly, we endow eMCSs with expressive bridge rules which allow the specification of how contexts should react and evolve. The resulting system will be equipped with a semantics based on the novel
The new eMCs are in line with the broad motivation presented in [6], and share some of the features with the framework of reactive Multi-Context Systems (mMCS) sketched in [6, 11]. However, some differences set them apart—which will become more apparent after we present eMCs—namely regarding how observations are handled, and also the kind of state transitions that can be made, with implications in the fulfillment of the requirements for this kind of systems.

Example 1 (Running example) Throughout this paper, we will illustrate some concepts using the scenario of an internet forum. As usual, users are divided into categories, and this division influences their permissions to post messages. In such a scenario, it is natural that knowledge is distributed among several contexts and we consider three: a local database which contains all information about existing users, topics, and posts; an ontology context, meant to be a previously existing general ontology that provides a comprehensive model to represent online communities and related user-generated content; finally, a context to model login and permission policy rules.

2 Preliminaries

Following [7], a multi-context system (MCS) consists of a collection of components, each of which contains knowledge represented in some logic, defined as a triple $L = (KB, BS, ACC)$ where $KB$ is the set of well-formed knowledge bases of $L$, $BS$ is the set of possible belief sets, and $ACC : KB → 2^{BS}$ 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 $KB$ and $BS$ is a set. We also define $F_L = \{s : s ∈ KB ∧ s ∈ KB_L\}$.

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. Yet, bridge rules in MCSs only allow adding information to the knowledge base of their corresponding context.

In [8], an extension, called managed Multi-Context Systems (mMCs), 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 $F_L^{OP} = \{op(s) : op ∈ OP ∧ s ∈ F_L\}$ be the set of operational formulas that can be built from $OP$ and $F_L$. 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 $OP$, $mng : 2^{F_L^{OP}} × KB → (2^{KB} \setminus \{∅\})$, 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 $mng(∅, kb) = \{kb\}$. Now, for a sequence of logics $L = \{L_1, ..., L_n\}$ and a management base $OP$, an $L_i$-bridge rule $σ$ over $L, 1 ≤ i ≤ n$, is of the form $H(σ) ← B(σ)$ where $H(σ) ∈ F_{L_i}^{OP}$, and $B(σ)$ is a set of bridge literals of the forms $r : b$ and $not(r : b), 1 ≤ r ≤ n$, with $b$ a belief formula of $L_r$.

A managed Multi-Context System (mMCs) is a sequence $M = (C_1, ..., C_n)$, where each $C_i, i ∈ \{1, ..., n\}$, called a managed context, is defined as $C_i = (L_i, kb_i, br_i, OP_i, mng_i)$ where $L_i = (KB_i, BS_i, ACC_i)$ is a logic, $kb_i ∈ KB_i$, $br_i$ is a set of $L_i$-bridge rules, $OP_i$ is a management base, $mng_i$ is a management function over $L_i$ and $OP_i$. For the sake of readability, we consider a slightly restricted version of mMCs where ACC is a function and not a set of functions as for logic suites [8].

Example 2 (Ctd.) We present a simplified configuration of the knowledge bases of the three contexts for the internet forum example and refer to [10] and [8] for the (standard) definitions of their logics. The knowledge base of the database (DB) context is the set $\{\text{RegUser}(Bob), \text{topic}(T), \text{Admin}(John)\}$, representing that Bob is a registered user, John an administrator and $T$ a topic. The knowledge base of the Description Logic (DL) [4] context is the set of DL axioms $\{\text{Admin} ⊆ \text{Mod}, \text{Mod} ⊆ \text{RegUser}\}$ encoding that every administrator is a moderator, which in turn is a registered user. The knowledge base of the logic programming (LP) [14] context is the program:

\begin{align*}
\text{canWrite}(x, t) &← \text{loggedIn}(x), \text{topic}(t), \text{not closed}(t) \\
\text{canClose}(x, t) &← \text{loggedIn}(x), \text{mod}(x), \text{topic}(t), \text{not closed}(t) \\
\text{sendSMS}(x) &← \text{failLogin}(x), \text{not SMSsent}(x) \\
\text{blocked}(x) &← \text{failLogin}(x), \text{SMSsent}(x)
\end{align*}

(1) and (2) define when an user can write a post and close a topic, respectively, (3) represents that an SMS should be sent to the user when the first failed login occurs. In case an SMS was already sent, a failed login causes the user to be blocked (4).

For an mMCs $M = (C_1, ..., C_n)$, a belief state of $M$ is a sequence $S = \{S_1, ..., S_n\}$ such that each $S_i$ is an element of $BS_i$. For a bridge literal $(r : b), S \models (r : b)$ if $b ∈ S_r$ and $S \models not (r : b)$ if $b ∈ S_r$; for a set of bridge literals $B, S \models B$ if $S \models B$ for all $L ∈ B$. We say that a bridge rule $σ$ of a context $C_i$ is applicable given a belief state $S$ of $M$ if $S$ satisfies $B(σ)$. We can then define $app_i(S)$, the set of heads of bridge rules of $C_i$ which are applicable in $S$, by setting $app_i(S) = \{H(σ) : σ ∈ br_i ∧ S \models B(σ)\}$.

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. Formally, a belief state $S = \{S_1, ..., S_n\}$ of an mMCs $M$ is an equilibrium of $M$ if, for every $1 ≤ i ≤ n, S_i ∈ ACC_i(kb)$ for some $kb ∈ mng_i(app_i(S), kb_i)$.

3 Evolving Multi-Context Systems

In this section, we introduce evolving Multi-Context Systems, which generalize mMCs to a dynamic scenario in which contexts are enabled to react to external observations and evolve. For that purpose, we consider that some of the contexts in the MCS become so-called observation contexts whose knowledge bases will be constantly changing over time according to the observations made, similar, e.g., to streams of data from sensors. For simplicity of presentation, we consider discrete steps in time here.

\[\text{Definition 1} \quad \text{Given a management base } OP \text{ and a logic } L, \text{ we define } eF_L^{OP}, \text{ the evolving operational language, as } eF_L^{OP} = F_L^{OP} ∪ \{next(op(s)) : op(s) ∈ F_L^{OP}\}.\]
We can now define evolving Multi-Context Systems.

Definition 2 An evolving Multi-Context System (eMCS) is a sequence \( M_e = \langle C_1, \ldots, C_n \rangle \), where 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
- \( L_i = \langle KB_i, BS_i, ACC_i \rangle \) is a logic
- \( kb_i \in KB_i \)
- \( br_i \) is a set of \( L_i \)-bridge rules s.t. \( H(\sigma) \in eF^{Op}_i \)
- \( Op_i \) is a management base
- \( mng_i \) is a management function over \( L_i \) and \( Op_i \).

As already outlined, evolving contexts can be divided into regular reasoning contexts and special observation contexts that are meant to process a stream of observations which ultimately enables the entire eMCS to react and evolve in the presence of incoming observations. To ease the reading and simplify notation, w.l.o.g., we assume that the first \( \ell \) contexts, \( 0 \leq \ell \leq n \), in the sequence \( \langle C_1, \ldots, C_n \rangle \) are observation contexts, and, whenever necessary, such an eMCS can be explicitly represented by \( \langle C^1_{\ell}, \ldots, C^\ell_{\ell}, C_{\ell+1}, \ldots, C_n \rangle \).

As for McMCSs, a belief state for \( M_e \) is a sequence \( S = \langle S_1, \ldots, S_n \rangle \) such that, for each \( 1 \leq i \leq n \), we have \( S_i \in BS_i \).

Recall that the heads of bridge rules in an eMCS are more expressive than in an mMCS, since they may be of two types: those that contain next and those that do not. The former affect the current state of the knowledge base with a non persistent effect, while the latter are used to produce the knowledge base at the subsequent state, with a persisting effect. Therefore, we distinguish these two subsets.

Definition 3 Let \( M_e = \langle C_1, \ldots, C_n \rangle \) be an eMCS and \( S \) a belief state for \( M_e \). Then, for each \( 1 \leq i \leq n \), consider the following sets:
- \( app_i^{next}(S) = \{ op(s) : next(op(s)) \in app_i(S) \} \)
- \( app_i^{now}(S) = \{ op(s) : op(s) \in app_i(S) \} \)

To achieve a change in the current state while making such change persist, we can use two bridge rules with identical body, one with and one without the next operator.

Example 3 (Ctd.) We now present the internet forum eMCS \( M_e = \langle C_1^f, C_2^f, C_3, C_4 \rangle \) composed of one observation context \( C_1^f \) and three reasoning contexts \( C_2^f, C_3 \) and \( C_4 \), corresponding to the DB, DL, and LP context, respectively, whose knowledge bases are given in Example 2. The knowledge base and belief set language of \( C_1^f \) is composed of all the ground instances of write\((x,t,p)\), register\((x)\), mkPrvt\((x,t)\), login\((x)\), logout\((x)\), and failLogin\((x)\). The function \( ACC_1 \) assigns \( K \) to every set \( K \) of knowledge base formulas and \( br_1 \) is empty. The DB context comprises the set of bridge rules br_2:

\[
\text{next}(\text{ins}(\text{RegUser}(x))) \leftarrow 1: \text{register}(x), \text{not} 2: \text{RegUser}(x) \\
\text{next}(\text{ins}(\text{HasReply}(t,p))) \leftarrow 1: \text{write}(x,t,p), 4: \text{canWrite}(x,t) \\
\text{next}(\text{ins}(\text{Closed}(t))) \leftarrow 1: \text{close}(x,t), 4: \text{canClose}(x,t)
\]

The rules express how \( kb_2 \) evolves in face of incoming observations and the permission policies in \( C_4 \) declaring when to add a new registered user, a new post to a topic, or closing a topic, where \( ins \) is the usual database insertion operation. The use of next ensures that the effects persist. The bridge rules of the DL context \( C_3 \) import information from \( C_2 \) to allow further inferences.

\[
\text{add}(\text{Admin}(x)) \leftarrow 2: \text{Admin}(x) \\
\text{add}(\text{Mod}(x)) \leftarrow 2: \text{Mod}(x) \\
\text{add}(\text{RegUser}(x)) \leftarrow 2: \text{RegUser}(x)
\]

Note that, since we do not want to duplicate information already in \( C_2 \), we only import it temporarily to \( C_3 \) without using the operator.

\[
\text{next}. \quad \text{The LP context contains the following bridge rules:} \\
\text{next}(\text{upd}(\text{loggedIn}(x))) \leftarrow 1: \text{login}(x), \text{not} 4: \text{loggedIn}(x) \\
\text{next}(\text{upd}(\text{not loggedIn}(x))) \leftarrow 1: \text{logout}(x), 4: \text{loggedIn}(x)
\]

where \( rule = \text{not canWrite}(x,t) \leftarrow \text{not mod}(x) \).

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

Definition 4 Let \( M_e = \langle C_1, \ldots, C_n \rangle \) be an eMCS. A belief state \( S = \langle S_1, \ldots, S_n \rangle \) for \( M_e \) is an 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 \( S_i \in ACC_i(kb) \).

To be able to assign meaning to an eMCS evolving over time, we introduce evolving belief states, which are sequences of belief states, each referring to a subsequent time instant.

Definition 5 Let \( M_e = \langle C_1, \ldots, C_n \rangle \) be an eMCS. An evolving belief state of size \( s \) for \( M_e \) is a sequence \( S_e = \langle S^1, \ldots, S^s \rangle \) where each \( S^j, 1 \leq j \leq s, \) is a belief state for \( M_e \).

To enable an eMCS to react to incoming observations and evolve, an observation sequence, defined next, has to be processed. The idea is that the knowledge bases of the observation contexts \( C^o_i \) change according to that sequence.

Definition 6 Let \( M_e = \langle C_1^o, \ldots, C^o_n, C_{\ell+1}, \ldots, C_n \rangle \) be an eMCS. An observation sequence for \( M_e \) is a sequence \( Obs = \langle O^1, \ldots, O^m \rangle \), such that, for each \( 1 \leq j \leq m \), \( O^j = \langle o^j_1, \ldots, o^j_\ell \rangle \) is an instant observation with \( o^j_\ell \in KB \), for each \( 1 \leq i \leq \ell \).

To be able to update the knowledge bases in the evolving contexts, we need one further notation. Given an evolving context \( C_i \) and \( k \in KB_i \), we denote by \( C_i[k] \) the evolving context in which \( kb_i \) is replaced by \( k \), i.e., \( C_i[k] = \langle L_i, kb_i, OP_i, mng_i \rangle \).

We can now define that certain evolving belief states are evolving equilibria of an eMCS \( M_e = \langle C_1^o, \ldots, C^o_n, C_{\ell+1}, \ldots, C_n \rangle \) given an observation sequence \( Obs = \langle O^1, \ldots, O^m \rangle \) for \( M_e \). The intuitive idea is that, given a evolving belief state \( S_e = \langle S^1, \ldots, S^s \rangle \) for \( M_e \), in order to check if \( S_e \) is an evolving equilibrium, we need to consider a sequence of eMCSs, \( M^1, \ldots, M^s \) (each with \( \ell \) observation contexts), representing a possible evolution of \( M_e \) according to the observations in \( Obs \), such that \( S^j \) is a (static) equilibrium of \( M^j \). The knowledge bases of the observation contexts in \( M^j \) are exactly their corresponding elements \( o^j_\ell \in O^j \). For each of the other contexts \( C_i, \ell + 1 \leq i \leq n \), its knowledge base in \( M^j \) is obtained from the one in \( M^{j-1} \) by applying the operations in \( app_i^{next}(S)^{-1} \).
Definition 7 Let $M_e = \langle C_0, \ldots, C_{i+1}, \ldots \rangle$ be an eMCS, $S_e = \langle S^1, \ldots, S^n \rangle$ an evolving belief state of size $s$ for $M_e$, and $\text{Obs} = \langle O^1, \ldots, O^m \rangle$ an observation sequence for $M_e$ such that $m \geq s$. Then, $S_e$ is an evolving equilibrium of size $s$ of $M_e$ given $\text{Obs}$ iff for each $1 \leq j \leq s$, $S^j = \langle S^{k_i}, \ldots, S^{k_s} \rangle$ where, for each $\ell + 1 \leq i \leq s$, $k_i$ is defined inductively as follows:

- $k_i = kb_i$
- $k_i+1 \in mng_i(app_i(next(S^i)), k_i)$

Note that $next$ in bridge rule heads of observation contexts are thus without any effect, in other words, observation contexts can indeed be understood as managed contexts whose knowledge base changes with each time instant.

Example 4 (Ctd.) Consider the observation sequence $\text{Obs} = \langle O^1, O^2, O^3 \rangle$ such that $o_1 = \{\text{register}(Anna), \text{failLogin}(Bob)\}$, $o_2 = \{\text{login}(Anna), \text{failLogin}(Bob), \text{mkPrvt}(John,T)\}$, and $o_3 = \{\text{write}(Anna,T), \text{login}(Bob)\}$. Then, an evolving equilibrium of size $3$ of $M_e$ given $\text{Obs}$ is the sequence $\Gamma = \langle S^1, S^2, S^3 \rangle$ such that, for each $1 \leq j \leq 3$, $S^j = \langle S_1(i), S_2(i), S_3(i) \rangle$. Since it is not feasible to present the entire $\Gamma$, we just highlight some interesting parts related to the evolution of the system. E.g., we have that $\text{sendSMS}(Bob) \in S_1$, $\text{RegUser}(Anna) \in S_2$, and $\text{SMSGsent}(Bob) \in S_2$; and $\text{blocked}(Bob), \text{loggedIn}(Anna) \in S_3$, but not $\text{CanWrite}(Anna,T) \in S_1$ since $T$ was made private in the previous time instant by admin John.

In Def. 7, the number of considered time instances of observations, $m$, is greater or equal to the size of the evolving belief state. The intuition is that an equilibrium may also be defined for a part of the observation sequence only. An immediate consequence is that any subsequence of an evolving equilibrium is an evolving equilibrium.

Proposition 1 Let $M_e = \langle C_1, \ldots, C_n \rangle$ be an eMCS and $\text{Obs} = \langle O^1, \ldots, O^m \rangle$ an observation sequence for $M_e$. If $S_e = \langle S^1, \ldots, S^n \rangle$ is an evolving equilibrium of size $s$ of $M_e$ given $\text{Obs}$, then, for each $1 \leq j \leq s$, and every $1 \leq k \leq m$, we have that $(S^j_1, \ldots, S^j_k)$ is an evolving equilibrium of size $j$ of $M_e$ given the observation sequence $(O^1, \ldots, O^k)$.

It is not hard to see that an mMCS is a particular case of an eMCS with no observation context and whose bridge rule heads do not contain the operator $next$. Note that, since there are no observation contexts, an observation sequence for an mMCS is necessarily a sequence of empty instant observations. We prove the following result.

Proposition 2 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 $(S)$ is an evolving equilibrium of size $1$ of $M$ for some observation sequence $\text{Obs}$ of $M$ of size at least $1$.

We now define an operator that incrementally constructs the set of evolving equilibria of an eMCS, i.e., it constructs the set of evolving equilibria of size $n$ from the set of evolving equilibria of size $n-1$. Formally, given an eMCS $M_e = \langle C_1, \ldots, C_n, O_1, \ldots, O_i \rangle$ and an observation sequence $\text{Obs} = \langle O^1, \ldots, O^m \rangle$ for $M_e$, we define an operator $\Gamma$ on the set of evolving belief sets for $M_e$ of size at most $m$. First, given an evolving belief state $S_e = \langle S^1, \ldots, S^n \rangle$ with $s \leq m$, we define the set $Tr(S_e)$ of its traces, which describes how the reasoning contexts evolve:

$$\text{Tr}(S_e) = \{\langle K^1, \ldots, K^n \rangle | \text{each } K^j = \{k^j_1, \ldots, k^j_s\} \text{ and, for each } \ell + 1 \leq i \leq s, k^j_i \text{ is defined inductively as: } k^j_i = kb_i \text{ and } k^j_{i+1} \in mng_i(app_i(next(S^i)), k^j_i)\}$$

Note that $next$ in bridge rule heads of observation contexts are thus without any effect, in other words, observation contexts can indeed be understood as managed contexts whose knowledge base changes with each time instant.

Example 4 (Ctd.) Consider the observation sequence $\text{Obs} = \langle O^1, O^2, O^3 \rangle$ such that $o_1 = \{\text{register}(Anna), \text{failLogin}(Bob)\}$, $o_2 = \{\text{login}(Anna), \text{failLogin}(Bob), \text{mkPrvt}(John,T)\}$, and $o_3 = \{\text{write}(Anna,T), \text{login}(Bob)\}$. Then, an evolving equilibrium of size $3$ of $M_e$ given $\text{Obs}$ is the sequence $\Gamma = \langle S^1, S^2, S^3 \rangle$ such that, for each $1 \leq j \leq 3$, $S^j = \langle S_1(i), S_2(i), S_3(i) \rangle$. Since it is not feasible to present the entire $\Gamma$, we just highlight some interesting parts related to the evolution of the system. E.g., we have that $\text{sendSMS}(Bob) \in S_1$, $\text{RegUser}(Anna) \in S_2$, and $\text{SMSGsent}(Bob) \in S_2$; and $\text{blocked}(Bob), \text{loggedIn}(Anna) \in S_3$, but not $\text{CanWrite}(Anna,T) \in S_1$ since $T$ was made private in the previous time instant by admin John.

4 Inconsistency Management

Inconsistency management is an important topic for frameworks that aim at integrating knowledge from different sources, and this is all the more true when knowledge changes over time.

For the case of mMCSs, three forms of inconsistency are considered: nonexistence of equilibria, local inconsistency, and operator inconsistency [8]. The first has been extensively studied for MCCSs [10] 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 handling conflicts between operations in the heads of bridge rules. Since the latter is tightly connected to the management function, which is also crucial for dealing with local inconsistency [8], we only consider global and local inconsistency, generalize related concepts of [8] and transfer them to eMCSs.

We start by introducing two notions of (global) consistency differing only on which observation(s) to consider.

Definition 8 Let $M_e = \langle C_1, \ldots, C_n \rangle$ be an eMCS and $\text{Obs} = \langle O^1, \ldots, O^m \rangle$ an observation sequence for $M_e$. Then, $M_e$ is consistent with respect to $\text{Obs}$ if it has an evolving equilibrium of size $m$ given $\text{Obs}$, and strongly consistent if, for every observation sequence $\text{Obs}$ for $M_e$, $M_e$ is consistent with respect to $\text{Obs}$.

From Prop. 1, we immediately obtain that if there is subsequence of $\text{Obs}$ such that the considered eMCS is inconsistent, then the eMCS is also inconsistent for the entire sequence (and vice-versa).

Corollary 1 Let $M_e = \langle C_1, \ldots, C_n \rangle$ be an eMCS and $\text{Obs} = \langle O^1, \ldots, O^m \rangle$ an observation sequence for $M_e$. Then, $M_e$ is consistent w.r.t. $\text{Obs}$, iff $M_e$ is consistent w.r.t. $\langle O^1, \ldots, O^j \rangle$ for every $1 \leq j \leq m$.

It is obvious that strong consistency implies consistency w.r.t. any observation sequence, but not vice-versa, and that Corollary 1 can also be adapted for strong consistency. Unfortunately, verifying strong consistency is highly complex since it requires checking all possible observation sequences. Still, strong consistency is an important property if we want to ensure that an eMCS always has an
evolving equilibrium independently of the considered observation sequence, which is why we now establish conditions that ensure that an eMCS is strongly consistent (and thus consistent) and at the same time discuss some notions on inconsistency management.

Inconsistency management was discussed in [10] for MCSs based on the notions of diagnoses and explanations. Diagnosis aims at finding modifications on the bridge rules such that consistency is restored. Dually, explanations look for bridge rules that avoid/cause inconsistencies. Both notions were generalized to mMCSs in [8]. These notions can also be straightforwardly generalized to eMCSs, yielding evolving sequences of sets of diagnoses and explanations, respectively, one for each time instant. In general, these sets in such a sequence differ from one instant to another, thus not allowing one unique diagnosis or explanation. Those diagnoses and explanations that persist in each such set possibly indicate a more general structural problem in the eMCS. We leave the technical details, including an adaptation of Prop. 4 from [8], for an extended version, and focus only on two notions sufficient to ensure (strong) consistency.

The first one is that each context always has at least one acceptable belief set independently of the applicable operational formulas. Formally, a context \(C_i\) with \(kb_i\) in an eMCS \(M_e\) is \emph{totally coherent} iff, for every \(kb \in KB_i\), \(\text{ACC}_i(kb) \neq \emptyset\). The second one describes cycles between contexts that may cause inconsistency. Given an eMCS \(M_e = \langle C_1, \ldots, C_n \rangle\), we say \(\text{ref}_{i,j}(r)\) iff \(r\) is a bridge rule of context \(C_i\) and \((j : p)\) occurs in the body of \(r\). For an eMCS \(M_e\) and \(\{r_1, \ldots, r_k\} \subseteq \bigcup br_i\), we say that \(\{r_1, \ldots, r_k\}\) forms a cycle iff \(\text{ref}_{r_1}(i_1, i_2), \ldots, \text{ref}_{r_{k-1}}(i_{k-1}, i_k), \text{and } \text{ref}_{r_k}(i_k, i_1)\) hold. Then, \(M_e\) is acyclic if no such cycles exist. We can show the following.

**Proposition 3** Any acyclic eMCS with totally coherent contexts is strongly consistent.

A similar property holds for consistent mMCSs, which indicates that the extension to eMCSs 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 extends to eMCSs with incoming observations that also should be subject to consistency management. As in [8], we need to assume that each context has a notion of inconsistent belief state, which usually exists or is easily definable. This allows us to introduce the following notions. A knowledge base \(kb_i \in KB_i\) of a context \(C_i\) is said to be consistent if \(\text{ACC}_i(kb_i)\) does not contain an inconsistent belief set. A management function \(\text{mng}_i\), of a context \(C_i\) is said to be \emph{locally consistency preserving (lc-preserving)}, if for every set \(Op_i \subseteq F^P_i\) and consistent knowledge base \(kb_i \in KB_i\), we have that every element of \(\text{mng}_i(Op_i, kb_i)\) is a consistent knowledge base.

**Definition 9** Let \(M_e\) be an eMCS and \(Obs\) an observation sequence for \(M_e\). Then, \(M_e\) is 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.

Note that we do not consider a strong notion of local consistency, since this would require investigating properties of concrete management functions, which we leave for future work.

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 eMCS inconsistent.

**Proposition 4** Let \(M_e = \langle C_1, \ldots, C_n \rangle\) be an eMCS s.t. for each \(C_i, kb_i\) is consistent and \(\text{mng}_i\) lc-preserving. If \(Obs = \langle O^1, \ldots, O^m \rangle\) is an observation sequence for \(M_e\) s.t. each \(O^i\) is a consistent knowledge base, then \(M_e\) is locally consistent w.r.t. \(Obs\).

### 5 Complexity

The computational complexity of MCSs and mMCSs has been studied with a focus on existence of equilibria [7, 8] and inconsistency analysis [10]. Here, we consider the existence of an evolving equilibrium of size \(s\) for an eMCS \(M_e\) given an observation sequence \(Obs\), i.e., we check whether \(M_e\) is consistent, denoted \(\text{CONS}_M(M)\).

Using the operator \(\Gamma\) for that purpose would not be efficient, since it would, in general, compute all exponentially many evolving equilibria. Still, what the iteration of \(\Gamma\) clearly shows is that the computation of one evolving equilibrium of size \(s\) is simply a sequence of \(s\) computations – one for each time instant – which is why we can divide the problem and rely on notions previously developed.

For analyzing the complexity in each time instant, we can utilize output-projected belief states [10]. The idea is to consider only those beliefs that appear in some bridge rule body. Formally, given an evolving context \(C_i\) within \(M_e = \langle C_1, \ldots, C_n \rangle\), we can define \(\text{OUT}_i\) to be the set of all beliefs of \(C_i\) occurring in the body of some bridge rule in \(M_e\). The output-projection of a belief state \(S = \langle S_1, \ldots, S_n \rangle\) of \(M_e\) is the belief state \(S' = \langle S'_1, \ldots, S'_n \rangle\), \(S'_i = S_i \cap \text{OUT}_i\), for \(1 \leq i \leq n\). We obtain the following result for (static) equilibria which is adapted from the case of mMCSs [8].

**Proposition 5** An eMCS \(M_e = \langle C_1, \ldots, C_n \rangle\) has an equilibrium iff some output-projected belief state \(S' = \langle S'_1, \ldots, S'_n \rangle\) exists such that, for all \(1 \leq i \leq n\), \(S'_i \in \langle S_i \cap \text{OUT}_i : S_i \in \text{ACC}_i(kb_i) \land kb'_i \in \text{mng}_i(app^\text{now}(S'), kb_i)\rangle\).

Following [10, 8], the context complexity of \(C_i\) is the complexity of the following problem:

1. **(CC)** Decide, given \(Op_i \subseteq F^P_i\) and \(S'_i \subseteq \text{OUT}_i\), if exist \(kb'_i \in \text{mng}_i(Op_i, kb_i)\) and \(S_i \in \text{ACC}_i(kb'_i)\) s.t. \(S'_i = S_i \cap \text{OUT}_i\).

Note that \(C_i\) is explicitly represented by \(kb_i\) and \(br_i\), and the logic is implicit, i.e., existence of \(S_i\) is decided by an oracle. The context complexity \(\text{CC}(M)\) of an eMCS \(M_e\) is a (smallest) upper bound for the context complexity classes of all \(C_i\) [8].

Problem (CC) can intuitively be divided into two subproblems:

1. **(MC)** Compute some \(kb'_i \in \text{mng}_i(Op_i, kb_i)\) and (EC) decide whether \(S_i \in \text{ACC}_i(kb'_i)\) exists s.t. \(S'_i = S_i \cap \text{OUT}_i\), but, as argued in [8], considering (CC) suffices for complexity considerations.

Now, checking whether \(M_e\) is consistent, essentially amounts to guessing an evolving belief state of size \(s\) and then checking, for each of the \(s\) time instants, (MC) and (EC) ignoring all elements in \(Op_i\) with next, and additionally an independent (MC) ignoring all elements \(Op_i\) without next. Thus, we limit our considerations to (CC) and in dependence on \(\text{CC}(M)\), we show in the following table the complexity of \(\text{CONS}_M(M)\) for several complexity classes used in [8], where \(i \geq 1\) and entries denote membership results, resp. completeness results if (CC) is hard for some \(C_i:\)

<table>
<thead>
<tr>
<th>(\text{CONS}_M(M))</th>
<th>(P)</th>
<th>(\Sigma^P_1)</th>
<th>(\Delta^P_1)</th>
<th>(\text{PSPACE})</th>
<th>(\text{EXPTIME})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{CC}(M))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that these results exactly correspond to those for mMCSs in [8] which means that the transition from mMCSs to eMCSs does not increase the worst case complexity for checking consistency. The reason is that checking for existence of an evolving equilibrium only adds factors two and \(s\) for computing (MC) independently twice, and for the size of the evolving equilibrium, respectively. A more fine-grained analysis would consider (MC) and (EC) separately, but we conjecture that any changes can be traced back to (MC) and (EC) and would also affect mMCSs. We leave such a study for future work.
6 Related and Future Work

Evolving Multi-Context Systems share some of the main ideas of reactive Multi-Context Systems sketched in [6, 11, 9] inasmuch as both aim at extending mMCs to cope with dynamic observations. Three main differences distinguish them. First, whereas mMCs rely on a sequence of observations, each independent from the previous ones, rMCs encode such sequences within the same observation contexts, with its elements being explicitly timestamped. This means that with rMCs it is perhaps easier to write bridge rules that refer, e.g., to specific sequences of observations, which in eMCs would require explicit timestamps and storing the observations in some context, although at the cost that rMCs need to deal with explicit time which adds an additional overhead. Second, since in rMCs the contexts resulting from the application of the management operations are the ones that are used in the subsequent state, difficulties may arise in separating non-persistent and persistent effects, for example, allowing an observation to override some fact in some context while the observation holds, but without changing the context itself – such separation is easily encodable in eMCs given the two kinds of bridge rules, i.e., with or without operator next. Finally, bridge rules with next allow for the specification of transitions based on the current state, such as the one encoded by the rule
\[
\text{assert} \leftarrow \text{not p},
\]
which do not seem possible in rMCs. Overall, these differences indicate that an interesting future direction would be to merge both approaches, exploring a combination of explicitly timestamped observations with the expressiveness provided by operator next.

Another framework that aims at modeling the dynamics of knowledge is that of evolving logic programs EVOLP [2] focusing on updates of generalized logic programs. It is possible to show that EVOLP can be seen as a particular case of eMCs, using the operator next to capture the operator assert of EVOLP. We leave the details for an extended version. Closely related to EVOLP, hence to eMCs, are the two frameworks of reactive ASP, one implemented as a solver oclingo [13] and one described in [6]. The system oclingo extends an ASP solver for handling external modules provided at runtime by a controller. The output of these external modules can be seen as the observations of EVOLP. Unlike the observations in EVOLP, which can be rules, external modules in oclingo are restricted to produce atoms so the evolving capabilities are more restricted. On the other hand, oclingo permits committing to a specific answer-set at each state, a feature that is not part of EVOLP, nor of eMCs. Reactive ASP as described in [6] can be seen as a more straightforward generalization of EVOLP where operations other than assert for self-updating a program are permitted. Given the above mentioned embedding of EVOLP in eMCs, and the fact that eMCs permit several (evolution) operations in the head of bridge rules, it is also not difficult to show that Reactive ASP as described in [6] can be captured by eMCs.

An important topic for future work is to study minimal change in eMCs. 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 a variable that non-deterministically takes one of two values at each time instant: minimal change could force a constant value.

The dynamics of eMCs is one kind of dynamics, but surely not the only one. Studying the dynamics of the bridge rules is also an important topic, to a great extent orthogonal to the current development. Another form of dynamics is to perform AGM style belief revision at the (semantic) level of the equilibria, as in Wang et al [20], though different since knowledge is not incorporated in the contexts.

We can also consider the generalization of the notions of minimal and grounded equilibria [7] to eMCs to avoid, e.g., self-supporting cycles caused by bridge rules, or the introduction of preferences to deal with the existence of several evolving equilibria of an eMCs.

Also interesting is to apply the ideas in this paper to study the dynamics of frameworks closely related to MCSs, such as [17, 16].

ACKNOWLEDGEMENTS

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/IEIA-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


