Towards Efficient Evolving Multi-Context Systems
(Preliminary Report)

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. Recently, evolving Multi-Context Systems (eMCSs) have been introduced as an extension of mMCSs that add the ability to both react to, and reason in the presence of commonly temporary dynamic observations, and evolve by incorporating new knowledge. However, the general complexity of such an expressive formalism may simply be too high in cases where huge amounts of information have to be processed within a limited short amount of time, or even instantaneously. In this paper, we investigate under which conditions eMCSs may scale in such situations and we show that such polynomial eMCSs can be applied in a practical use case.

1 Introduction

Multi-Context Systems (MCSs) were introduced in [7], building on the work in [16, 27], to address the need for a general framework that integrates knowledge bases expressed in heterogeneous KR formalisms. Intuitively, instead of designing a unifying language (see e.g., [17, 26], and [23] with its reasoner NoHR [22]) 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 (cf. [1, 21, 24] and references therein for further motivation on hybrid languages and their connection to MCSs).

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 its head to the context’s knowledge base provided the queries (to other contexts) in the body 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 applications 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 [14, 13], C-SPARQL [5], Ontology Streams [25] 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.

To address this issue, two recent frameworks, evolving Multi-Context Systems (eMCSs) [19] and reactive Multi-Context Systems (rMCSs) [6, 12, 9] 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 (see related work in Sec. [9], the definition of eMCSs is presented in a more general way. That, however, means that, as shown in [19], the worst-case complexity is in general high, which may be problematic in dynamic scenarios where the overall system needs to evolve and react interactively. This is all the more true for huge amounts of data – for example raw sensor data is likely to be constantly produced in large quantities – and systems that are capable of processing and reasoning with such data are required.

At the same time, eMCSs inherit from MCSs the property that models, i.e., equilibria, may be non-minimal, which potentially admits that certain pieces of information are considered true based solely on self-justification. As argued in [7], minimality may not always be desired, which can in principle be solved by indicating for each context whether it requires minimality or not. Yet, avoiding self-justifications for those contexts where minimality is desired has not been considered in eMCSs.

In this paper, we tackle these problems and, in particular, consider under which conditions reasoning with evolving Multi-Context Systems can be done in polynomial time. For that purpose, we base our work on a number of notions studied in the context of MCSs that solve these problems in this case [7]. Namely, we adapt the notions of minimal and grounded equilibria to eMCSs, and subsequently a well-founded semantics, which indeed paves the way to the desired result.

The remainder of this paper is structured as follows. After introducing the main concepts regarding mMCSs in Sect. [2] in Sect. [3] we recall with more detail the framework of eMCSs already introducing adjustments to achieve polynomial reasoning. Then, in Sect. [4] we present an example use case, before we adapt and generalize notions from MCSs in Sect. [5] as outlined. We conclude in Sect. [6] with discussing related work and possible future directions.

1 CENTRIA & Departamento de Informática, Faculdade Ciências e Tecnologia, Universidade Nova de Lisboa, email: rjrg@fct.unl.pt
2 Preliminaries: Managed Multi-Context Systems

Following [2], 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 \( \rightarrow \) \( 2^{BS} \) 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, and define \( F = \{ s : s \in kb \wedge kb \in KB \} \).

In addition to the knowledge base in each component, bridge rules 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 an edge to assert in one component given certain beliefs held in the other component(s). Bridge rules in MCSs only allow adding an edge to assert in one component given certain beliefs held in the other component(s). Bridge rules in MCSs only allow adding an edge to assert in one component given certain beliefs held in the other component(s).

In [8], an extension, 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 that can be built from \( OP \) and \( F \). 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 \( OP \), \( mng : 2^{OF} \times KB \rightarrow KB \), i.e., \( mng(op, kb) \) is the knowledge base that results from applying the operations in \( op \) to the knowledge base \( kb \).

Note that this is already a specific restriction in our case, as \( mng \) commonly returns a (non-empty) set of possible knowledge bases for mMCSs (and eMCSs). We also assume that \( mng(\emptyset, kb) = kb \).

Now, for a sequence of logics \( L = \langle L_1, \ldots, L_n \rangle \) and a management base \( OP_i \), an \( L_i \)-bridge rule \( \sigma \) over \( L \), \( 1 \leq i \leq n \), is of the form \( H(\sigma) \leftarrow B(\sigma) \) where \( H(\sigma) \in OF \), and \( B(\sigma) \) is a set of bridge literals of the forms \( (r : b) \) and \( \text{not}(r : b) \), \( 1 \leq r \leq n \), with \( b \) a belief formula of \( L_r \).

A managed Multi-Context System (mMCS) is a sequence \( M = \langle C_1, \ldots, C_n \rangle \), where each \( C_i \), \( i \in \{1, \ldots, n\} \), is a context, defined as \( C_i = \langle L_i, kb_i, br_i, OP_i, mng_i \rangle \). The evolving operational language, as

3 Evolving Multi-Context Systems

In this section, we recall evolving Multi-Context Systems as introduced in [19] including some alterations that are in line with our intentions to achieve polynomial reasoning. As indicated in [19], 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.

The changing observations then 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 Given a management base \( OP \) and a logic \( L \), we define \( eOF \), the evolving operational language, as \( eOF = OF \cup \{ \text{next}(op(s)) : op(s) \in OF \} \).

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:

- \( L_i = (KB_i, BS_i, ACC_i) \) is a logic
- \( kb_i \in KB_i \)
- \( br_i \) is a set of \( L_i \)-bridge rules s.t. \( H(\sigma) \in eOF \)
- \( OP_i \) 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 \( \langle C_{\ell+1}, \ldots, C_n, C_1, \ldots, C_\ell \rangle \).

As for mMCSs, 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 \( \text{next} \) 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.

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:

- \( \text{app}_{\text{next}}(S) = \{ op(s) : \text{next}(op(s)) \in \text{app}(S) \} \)
- \( \text{app}_{\text{next}}(S) = \{ op(s) : op(s) \in \text{app}(S) \} \)

Note that if we want an effect to be instantaneous and persistent, then this can also be achieved using two bridge rules with identical body, one with and one without \( \text{next} \) in the head.

Similar to equilibria in mMCS, the (static) equilibrium is defined to incorporate instantaneous effects based on \( \text{app}_{\text{next}}(S) \) alone.
The idea is that the knowledge bases of the observation contexts an observation sequence defined in the following has to be processed. 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 \(S_e = \langle S^1, \ldots, S^n \rangle\) 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 in the following has to be processed. The idea is that the knowledge bases of the observation contexts \(C_i\) change according to that sequence.

\(\text{Definition 6}\) Let \(M_e = \langle C_1, \ldots, C_n \rangle\) be an eMCS. An observation sequence for \(M_e\) is a sequence \(\text{Obs} = \langle O^1, \ldots, O^m \rangle\), such that, for each \(1 \leq j \leq m\), \(O^j = \langle \ell_j, \ldots, \ell_{j+\ell} \rangle\) is an instant observation with \(\ell_j \in KB\), 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 \(k \in KB\), we denote by \(C_i[k]\) the evolving context in which \(k\) is replaced by \(k\), i.e., \(C_i[k] = \langle L_i, k, br, OP, \text{mnq}, \text{app} k^\text{next} (S^i) \rangle\).

We can now define that certain evolving belief states are evolving equilibria of an eMCS \(M_e = \langle C_1, \ldots, C_n \rangle\) given an observation sequence \(\text{Obs} = \langle O^1, \ldots, O^m \rangle\) for \(M_e\). The intuitive idea is that, given an evolving belief state \(S_e = \langle S^1, \ldots, S^n \rangle\) for \(M_e\), in order to check if \(S_e\) is an evolving equilibrium, we need to consider a sequence of eMCSs, \(M^1, \ldots, M^\ell\) (each with \(\ell\) observation contexts), representing a possible evolution of \(M_e\) according to the observations in \(\text{Obs}\), such that \(S^n\) is a (static) equilibrium of \(M^\ell\). The knowledge bases of the observation contexts in \(M^\ell\) are exactly their corresponding elements \(\ell_i \in O^{\ell}\). For each of the other contexts \(C_i\), \(\ell + 1 \leq i \leq n\), its knowledge base in \(M^\ell\) is obtained from the one in \(M^{\ell-1}\) by applying the operations in \(\text{app} k^\text{next} (S^i)\).

\(\text{Definition 7}\) Let \(M_e = \langle C_1, \ldots, C_n \rangle\) 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 \(s\) for \(M_e\), given \(\text{Obs}\) iff, for each \(1 \leq j \leq s\), \(S^j\) is an equilibrium of \(M^j\) = \(\langle C^1[\ell_1^j], \ldots, C^n[\ell_n^j], C_{e+1}[k_{e+1}^j], \ldots, C_n[k_n^j] \rangle\) where, for each \(\ell + 1 \leq i \leq n\), \(k_i^j\) is defined inductively as follows:

\(k_1^j = k_b\)
\(k_i^j + 1 = \text{mnq}(\text{app} k^\text{next}(S^i), k_i^j)\)

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.

The essential difference to \([19]\) is that the \(k_i^j + 1\) can be effectively computed (instead of picking one of several options), simply because \(\text{mnq}\) always returns one knowledge base. The same applies in Def \([1]\).

As shown in \([19]\), two consequences of the previous definitions are that any subsequence of an evolving equilibrium is also an evolving equilibrium, and mMCSs are a particular case of eMCSs.

\(\text{4 Use Case Scenario}\)

In this section, we illustrate eMCSs adapting a scenario on cargo shipment assessment taken from \([22]\).

The customs service for any developed country assesses imported cargo for a variety of risk factors including terrorism, narcotics, food and consumer safety, pest infestation, tariff violations, and intellectual property rights.\(^3\) Assessing this risk, even at a preliminary level, involves extensive knowledge about commodities, business entities, trade patterns, government policies and trade agreements. Some of this knowledge may be external to a given customs agency: for instance the broad classification of commodities according to the international Harmonized Tariff System (HTS), or international trade agreements. Other knowledge may be internal to a customs agency, such as lists of suspected violators or of importers who have a history of good compliance with regulations. While some of this knowledge is relatively stable, much of it changes rapidly. Changes are made not only at a specific level, such as knowledge about the expected arrival date of a shipment; but at a more general level as well. For instance, while the broad HTS code for tomatoes (0702) does not change, the full classification and tariffs for cherry tomatoes for import into the US changes seasonally.

Here, we consider an eMCS \(M_e = \langle C_1, C_2, C_3, C_4 \rangle\) composed of two observation contexts \(C_1^e\) and \(C_2^e\), and two reasoning contexts \(C_3\) and \(C_4\). The first observation context is used to capture the data of passing shipments, i.e., the country of their origination, the commodity they contain, their importers and producers. Thus, the knowledge base and belief set language of \(C_1^e\) is composed of all the ground atoms over ShpmtCommod, ShpmtDeciHTSCde, ShpmtImporter, ShpmtCountry, ShpmtProducer, and also GrapeTomato/1 and CherryTomato/1. The second observation context \(C_2^e\) serves to insert administrative information and data from other institutions. Its knowledge base and belief set language is composed of all the ground atoms over NewEUMember/1, Mising/1, and RandomInspection/1. Neither of the two observation contexts has any bridge rules.

The reasoning context \(C_3\) is an ontological Description Logic (DL) context that contains a geographic classification, along with information about producers who are located in various countries. It also contains a classification of commodities based on their harmonized tariff information (HTS chapters, headings and codes, cf. \url{http://www.wto.org/tata/hts}). We refer to \([11]\) and \([8]\) for the standard definition of \(L_s; k_b\) is given as follows:

\[
\text{Commodity} \equiv (\text{HTSCode.T})
\]
\[
\text{EdibleVegetable} \equiv (\text{HTSChapter.} \{'07\'})
\]
\[
\text{CherryTomato} \equiv (\text{HTSCode.} \{'07020020\'})
\]
\[
\text{Tomato} \equiv (\text{HTSHeading.} \{'0702\'})
\]
\[
\text{GrapeTomato} \equiv (\text{HTSCode.} \{'07020010\'})
\]
\[
\text{CherryTomato} \sqsubseteq \text{Tomato}
\]
\[
\text{GrapeTomato} \sqsubseteq \text{EdibleVegetable}
\]
\[
\text{EURegisteredProducer} \equiv (\text{RegisteredProducer.EUCountry})
\]
\[
\text{LowRiskEUCommodity} \equiv (\text{ExpeditableImporter.T})\sqcap
\]
add(CherryTomato(x)) ← (1: CherryTomato(x))
add(GrapeTomato(x)) ← (1: GrapeTomato(x))
next(add(EUCountry(x))) ← (2: NewEUMember(x))
add(CommodCountry(x, y)) ← (1: ShpmtCommod(x, z)), (1: ShpmtCountry(z, y))
add(ApprovedImportOf(y, x))

Note that \( kb_3 \) can indeed be expressed in the DL \( EL^{++} \) [4] for which standard reasoning tasks, such as subsumption, can be computed in PTIME.

Finally, \( C_4 \) is a logic programming (LP) indicating information about importers, and about whether to inspect a shipment either to check for compliance of tariff information or for food safety issues. For \( L_4 \) we consider that KB, the set of normal logic programs over a signature \( \Sigma \), is the set of atoms over \( \Sigma \), and ACC, \( (kb) \) returns returns a singleton set containing only the set of true atoms in the unique well-founded model. The latter is a bit unconventional, since this way undefinedness under the well-founded semantics [15] is merged with false information. However, as long as no loops over negation occur in the LP context (in combination with its bridge rules), undefiniteness does not occur, and the obvious benefit of this choice is that computing the well-founded model is PTIME-complete [10]. We consider \( OP_4 = OP_3 \), and \( kb_4 \) and \( br_4 \) are given as follows:

AdmissibleImportOf(x) ← ~SuspectedBadGuy(x).
PartialInspection(x) ← RandomInspection(x).
FullInspection(x) ← ~CompliantShpmt(x).
SuspectedBadGuy(y).
next((SuspectedBadGuy(x)) ← (2: Misfiling(x))
add(ApprovedImportOf(y, x)) ← (3: EdibleVegetable(x))
add(ApprovedImportOf(y, x)) ← (1: GrapeTomato(x))
add(CompliantShpmt(x)) ← (1: ShpmtCommod(x, y)), (3: HTSCode(y, z)), (1: ShpmtDeclHTSCode(x, z))
add(RandomInspection(x)) ← (1: ShpmtCommod(x, y)), (2: Random(y))
add(PartialInspection(x)) ← (1: ShpmtCommod(x, y)), not (3: LowRiskEUCommodity(y))
add(FullInspection(x)) ← (1: ShpmtCommod(x, y)), (3: Tomato(y)), (1: ShpmtCountry(x, slovakia))

Now consider the observation sequence \( Obs = (O^1, O^2, O^3) \) where \( c_1 \) consists of the following atoms on \( s_1 \) (where \( s \) in \( s_1 \) stands for shipment, \( e \) for commodity, and \( i \) for importer):

ShpmtCommod(s_1, c_1) ShpmtDeclHTSCode(s_1, ’07020010’)
ShpmtImporter(s_1, i_1) CherryTomato(c_1)
\( a_1^2 \) of the following atoms on \( s_2 \):

ShpmtCommod(s_2, c_2) ShpmtDeclHTSCode(s_2, ’07020020’)
ShpmtImporter(s_2, i_2) ShpmtCountry(s_2, portugal)
CherryTomato(c_2)
and \( a_1^1 \) of the following atoms on \( s_3 \):

ShpmtCommod(s_3, c_3) ShpmtDeclHTSCode(s_3, ’07020010’)
ShpmtImporter(s_2, i_3) ShpmtCountry(s_2, portugal)
GrapeTomato(c_3) ShpmtProducer(s_3, p_1)

while \( a_2^1 = a_2^2 = \emptyset \) and \( a_2^1 = \{ Misfiling(t_3) \} \). Then, an evolving equilibrium of size 3 of \( M_e \) is given \( Obs \) is the sequence \( S_e = (S^1, S^2, S^3) \) such that, for each \( 1 \leq j \leq 3, S^j = (S^j_1, S^j_2, S^j_3) \). Since it is not feasible to present the entire \( S_e \) we just highlight some interesting parts related to the evolution of the system, i.e., we have that FullInspection(\( s_1 \)) \( s_1 \) since the HTS code does not correspond to the cargo; no inspection on \( s_2 \) since the shipment is compliant, \( c_2 \) is an EU commodity, and \( s_2 \) was not picked for random inspection; and PartialInspection(\( s_1 \)) \( s_1 \), even though \( s_3 \) comes from a EU country, because \( s_3 \) has been identified at time instant 2 for misfiling, which has become permanent info available at time 3.

5 Grounded Equilibria and Well-founded Semantics

Even if we only consider MCSs \( M \), which are static and where an implicit \( mng \) always returns precisely one knowledge base, such that reasoning in all contexts can be done in PTIME, then deciding whether \( M \) has an equilibrium is in NP [7][8]. The same result necessarily also holds for eMCSs, which can also be obtained from the considerations on eMCSs [19].

A number of special notions were studied in the context of MCSs that tackle this problem [7]. In fact, the notion of minimal equilibria was introduced with the aim of avoiding potential self-justifications. Then, grounded equilibria as a special case for so-called reducible MCSs were presented for which the existence of minimal equilibria can be effectively checked. Subsequently, a well-founded semantics for such reducible MCSs was defined under which an approximation of all grounded equilibria can be computed more efficiently. In the following, we transfer these notions from static MCSs in [7] to dynamic eMCSs and discuss under which (non-trivial) conditions they can actually be applied.

Given an eMCS \( M_e = (C_1, \ldots, C_n) \), we say that a static equilibrium \( S = (S_1, \ldots, S_n) \) is minimal if there is no equilibrium \( S’ = (S_1, \ldots, S_n) \) such that \( S_i \subseteq S_i’ \) for all \( i \) with \( 1 \leq i \leq n \) and \( S_j’ \subseteq S_j \) for some \( j \) with \( 1 \leq j \leq n \).

This notion of minimality ensures the avoidance of self-justifications in evolving equilibria. The problem with this notion in terms of computation is that such minimization in general adds an additional level in the polynomial hierarchy. Therefore, we now formalize conditions under which minimal equilibria can be effectively checked. The idea is that the grounded equilibrium will be assigned to an eMCS \( M_e \) if all the logics of all its contexts can be reduced to special monotonic ones using a so-called reduction function. In the case where the logics of all contexts in \( M_e \) turn out to be monotonic, the minimal equilibrium will be unique.

Formally, a logic \( L = (KB, BS, ACC) \) is monotonic if
1. \( ACC(kb) \) is a singleton set for each \( kb \in KB \), and
2. \( S \subseteq S’ \) whenever \( kb \subseteq kb’ \), \( ACC(kb) = \{ S \} \), and \( ACC(kb’ \} = \{ S’} \).

Furthermore, \( L = (KB, BS, ACC) \) is reducible if for some KB* \( \subseteq KB \) and some reduction function \( red : KB \times BS \to KB^* \):

1. the restriction of \( L \) to \( KB^* \) is monotonic,
2. for each \( kb \in KB \), and all \( S, S’ \in BS \):
   - \( red(kb, S) = kb \) whenever \( kb \in KB^* \),
   - \( red(kb, S) \subseteq red(kb, S’) \) whenever \( S’ \subseteq S \),
   - \( S \in ACC(kb) \) if \( ACC(red(kb, S)) = \{ S \} \).
Then, an evolving context $C = (L, kb, br, OP, mng)$ is reducible if its logic $L$ is reducible and, for all $op \in E^L_{OP}$ and all belief sets $S$, $\text{red}(mng(op, kb), S) = mng(op, \text{red}(kb, S))$.

An eMCS is reducible if all of its contexts are. Note that a context is reducible whenever its logic $L$ is monotonic. In this case $KB^*$ coincides with $KB$ and $\text{red}$ is the identity with respect to the first argument.

As pointed out in [7], reducibility is inspired by the reduce in (non-monotonic) answer set programming. The crucial and novel condition in our case is the one that essentially says that the reduction function $\text{red}$ and the management function $mng$ can be applied in an arbitrary order. This may restrict to some extent the sets of operations $OP$ and $mng$, but in our use case scenario in Sect. 4 all contexts are indeed reducible.

A particular case of reducible eMCSs, definite eMCSs, does not require the reduction function and admits the polynomial computation of minimal evolving equilibria as we will see next. Namely, a reducible eMCS $M_e = \langle C_1, \ldots, C_n \rangle$ is definite if

1. none of the bridge rules in any context contains not, and
2. for all $i$ and all $S \in \text{BS}_i$, $\text{kb}_i = \text{red}(\text{kb}_i, S)$.

In a definite eMCS, bridge rules are monotonic, and knowledge bases are already in reduced form. Inference is thus monotonic and a unique minimal equilibrium exists. We take this equilibrium to be a unique minimal equilibrium exists. We take this equilibrium to be

$\langle O_1, \ldots, O_{\text{app}} \rangle$.

Then, $\langle O_1, \ldots, O_{\text{app}} \rangle$ is the grounded equilibrium of $M_e$ (see Def. 8), denoted by $\text{GE}(M_e)$. The well-founded semantics for reducible eMCSs $M_e$ can be obtained by simply applying this operator. Hence, the Knaster-Tarski theorem, $\gamma_{M_e}(S) = \text{GE}(M_e)$, provided $BS_i$, for each logic $L_i$ in all the contexts of $M_e$ has a least element $S^\ast$. Such eMCSs are called normal.

The following result generalizes Proposition 2 from [7].

**Proposition 2** Every grounded equilibrium of a reducible eMCS $M_e$ such that all $OP_i$ are monotonic is a minimal equilibrium of $M_e$.

This can again be generalized to evolving grounded equilibria.

**Definition 11** Let $M_e = \langle C_1, \ldots, C_n \rangle$ be a normal, reducible eMCS such that all $OP_i$ are monotonic. Then $\gamma_{M_e}(S) = \text{GE}(M_e)$, provided $BS_i$, for each logic $L_i$ in all the contexts of $M_e$ has a least element $S^\ast$. Such eMCSs are called normal.

The following result can be straightforwardly adopted from [7].

**Proposition 3** Let $M_e = \langle C_1, \ldots, C_n \rangle$ be a reducible eMCS such that all $OP_i$ are monotonic. Then $\gamma_{M_e}$ is antimonotone.

As usual, applying $\gamma_{M_e}$ twice yields a monotonic operator. Hence, by the Knaster-Tarski theorem, $\gamma_{M_e}$ has a least fixedpoint which determines the well-founded semantics.

**Definition 12** Let $M_e = \langle C_1, \ldots, C_n \rangle$ be a normal, reducible eMCS such that all $OP_i$ are monotonic. Then $\gamma_{M_e} \circ \gamma_{M_e}$ is the least fixed point which determines the well-founded semantics $\gamma_{WFS}(M_e)$ of $M_e$.

Starting with the least state $S^\ast = \langle S_1^\ast, \ldots, S_n^\ast \rangle$, this fixedpoint can be iterated, and the following correspondence between $WFS(M_e)$ and the grounded equilibria of $M_e$ can be shown.

For evolving belief states $S_i$ of size $s$ and an observation sequence $\text{Obs}_i$ for $M_e$, this proposition yields that the evolving grounded equilibrium for definite eMCSs can be obtained by simply applying this iteration $s$ times.

Grounded equilibria for general eMCSs are defined based on a reduct which generalizes the Gelfond-Lifschitz reduct to the multi-context case:
equilibria, even though $\text{WFS}(M_i)$ may itself not necessarily be an equilibrium. Yet, if all $\text{ACC}$ deterministically return one element of $\text{BS}$, and the eMCS is acyclic (i.e., no cyclic dependencies over bridge rules exist between beliefs in the eMCS see [19]), then the grounded equilibrium is unique and identical to the well-founded semantics. This is indeed the case for the use case in Sect.4.

As before, the well-founded semantics can be generalized to evolving belief states.

**Definition 13** Let $M_e = \langle C_1, \ldots, C_n \rangle$ be a normal, reducible eMCS such that all $O_P_i$ are monotonic, and $\text{Obs} = \langle O^1, \ldots, O^m \rangle$ an observation sequence for $M_e$ such that $m \geq s$. The evolving well-founded semantics of $M_e$, denoted $\text{WFS}_e(M)$, is the evolving belief state $S_e = \langle S_1, \ldots, S_n \rangle$ of size $s$ for $M_e$ such that $S_i$ is the well-founded semantics of $M_i$ defined as in Definition 2.

Finally, as intended, we can show that computing the evolving well-founded semantics of $M_e$ can be done in polynomial time under the restrictions established so far. For analyzing the complexity in each time instant, we can utilize output-projected belief states [11]. 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 $\text{OUT}$ to be the set of all beliefs of $C_i$ occurring in some bridge rule in $M_e$. The output-projection of a belief state $S_i = \langle S_1, \ldots, S_n \rangle$ of $M_e$ is the belief state $S'_i = \langle S'_1, \ldots, S'_n \rangle$, $S'_i = S_i \cap \text{OUT}_i$, for $1 \leq i \leq n$.

Following [11, 8], we can adapt the context complexity of $C_i$ from [19] as the complexity of the following problem:

**Theorem 1** Let $M_e = \langle C_1, \ldots, C_n \rangle$ be a normal, reducible eMCS such that all $O_P_i$ are monotonic, $\text{Obs} = \langle O^1, \ldots, O^m \rangle$ an observation sequence for $M_e$, and (CC) is in $\text{PTIME}$ for all $C_i$. Then, for $s \leq m$, computing $\text{WFS}_e(M_e)$ is in $\text{PTIME}$.

This, together with the observation that $\text{WFS}_e(M_e)$ coincides with the unique grounded equilibrium, allows us to verify that computing the results in our use case scenario can be done in polynomial time.

### 6 Related and Future Work

In this paper we have studied how eMCSs can be revised in such a way that polynomial reasoning is possible, and we have discussed an example use case to which this result applies. We have also investigated the adaptation of notions concerning minimality of (evolving) equilibria, and we observe that the notion of reducible eMCSs is considerably restricted, but not to the same extent as the efficient computation of the well-founded semantics requires. An open question is whether a more refined computation eventually tailored to less restrictive operations than considered here can be used to achieve similar results.

As mentioned in the Introduction, eMCSs share the main ideas of reactive Multi-Context Systems sketched in [6] [12] [9] inasmuch as both aim at extending mMCSs to cope with dynamic observations. Three main differences distinguish them. First, whereas eMCSs rely on a sequence of observations, each independent from the previous ones, rMCSs encode such sequences within the same observation contexts, with its elements being explicitly timestamped. This means that with rMCSs it is perhaps easier to write bridge rules that refer, e.g., to specific sequences of observations, which in eMCSs would require explicit timestamps and storing the observations in some context, although at the cost that rMCSs need to deal with explicit time which adds an additional overhead. Second, since in rMCSs 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 eMCSs given the two kinds of bridge rules, i.e., with or without operator $\text{next}$. Finally, bridge rules with $\text{next}$ allow for the specification of transitions based on the current state, such as the one encoded by the rule $\text{next}(\text{add}(p)) \leftarrow \neg p$, which do not seem possible in rMCSs. 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 $\text{next}$.

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 eMCSs, using the operator $\text{next}$ to capture the operator $\text{assert}$ 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 clingo [14] and one described in [6]. The system clingo 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 clingo are restricted to produce atoms so the evolving capabilities are very restricted. On the other hand, clingo 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 $\text{assert}$ for self-updating a program are permitted. Given the above mentioned embedding of EVOLP in eMCS, and the fact that eMCSs 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 eMCSs.

Also, as already outlined in [20], an important non-trivial topic is the study of the notion of minimal change within an evolving equilib-rium. 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 takes one of two values at each time instant: minimal change could force a constant value.

Another important issue open for future work is a more fine-grained characterization of updating bridge rules (and knowledge bases) as studied in [13] in light of the encountered difficulties when updating rules [23] [29] [31] and the combination of updates over various formalisms [29] [30].

Also interesting is to study how to perform AGM style belief revision at the (semantic) level of the equilibria, as in Wang et al [33], though different since knowledge is not incorporated in the contexts.
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/IEA-CC/012182/2010). 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


