Towards Inconsistency Management in Reactive Multi-Context Systems

Gerhard Brewka[1] and **Stefan Ellmauthaler**[1] and **Ricardo Gonçalves**[2] and **Matthias Knorr**[2] and **João Leite**[2] and **Jörg Pührer**[1]

Abstract. In this paper, we begin by introducing reactive multi-context systems (rMCSs), a framework for reactive reasoning in the presence of heterogeneous knowledge sources. In particular, we show how to integrate data streams into multi-context systems (MCSs) and how to model the dynamics of the systems, based on two types of bridge rules. We then discuss various methods for handling inconsistencies, a problem that occurs with reasoning based on multiple knowledge sources that need to be integrated, with a special focus on non-existence of equilibria.

1 Introduction

The wide and increasing availability of machine-processable data and knowledge – fueled by initiatives such as the Semantic Web, Linked Open Data, and the Internet of Things, among others – has prepared the ground and called for a new class of dynamic, rich, knowledge-intensive applications. Such new applications require automated reasoning based on the integration of several heterogeneous knowledge bases – possibly overlapping, independently developed, and written in distinct languages with different semantic assumptions – together with data/event streams produced by sensors and detectors, to support automation and problem-solving, to enforce traceable and correct decisions, and to facilitate the internalization of relevant dynamic data and knowledge.

Consider, for example, a scenario where Dave, an elderly person suffering from dementia, lives alone in an apartment equipped with various sensors, e.g., smoke detectors, cameras, and body sensors measuring relevant body functions (e.g., pulse, blood pressure, etc.). An assisted living application in such a scenario could leverage the information continuously received from the sensors, together with Dave's medical records stored in a relational database, a biomedical health ontology with information about diseases, their symptoms and treatments, represented in some description logic, some action policy rules represented as a non-monotonic logic program, to name only a few, and use it to detect relevant events, suggest appropriate action, and even raise alarms, while keeping a history of relevant events and Dave's medical records up to date, thus allowing him to live on his own despite his condition. For example, after detecting that Dave left the room while preparing a meal, the system could warn him about the situation in case he does not return soon. It could also even turn the stove off in case it detects that Dave fell asleep, not wanting to wake him up because his current treatment/health status values rest over immediate nutrition. Naturally, if Dave is not gone long enough, and no sensor shows any potential problems (smoke, gas, fire, etc.), then the system should seamlessly take no action. Another illustrative example would be a situation where the system observes that Dave's smart watch is indicating a high heart rate (tachycardia), in which case it would request Dave to measure his blood pressure (hypertension) – calling for assistance if Dave ignores the request – and, based on the measurements, determine whether to raise an alarm, calling an ambulance for example, or whether the readings are caused by some infection already under treatment or a decongestant that Dave recently took, in which case nothing would need to be done.

The requirements posed by novel applications such as the one just described, together with the availability of a vast number of knowledge bases – written using many different formalisms – and streams of data/events produced by sensors/detectors, has led modern research in knowledge representation and reasoning to face two fundamental problems: dealing with the **integration** of heterogeneous data and knowledge, and dealing with the **dynamics** of such novel knowledge based systems.

Integration The first problem stems from the availability of knowledge bases written in many different languages and formats developed over the last decades, from the rather basic ones, such as relational databases or the more recent triple-stores, to the more expressive ones, such as ontology languages (e.g., description logics), temporal and modal logics, non-monotonic logics, or logic programs under answer set semantics, to name just a few. Each of these formalisms was developed for different purposes and with different design goals in mind. Whereas some of these formalisms could be combined to a new, more expressive formalism, with features from its constituents – such as dl-programs [13] and Hybrid MKNF [31, 27] which, to different extent, combine description logics and logic programs under answer set semantics –, in general this is simply not feasible, either due to the mismatch between certain assumptions underlying their semantics, or because of the high price to pay, often in terms of complexity, sometimes even in terms of decidability. It is nowadays widely accepted that there simply is no such thing as a single universal, general purpose knowledge representation language.

What seems to be needed is a principled way of integrating knowledge expressed in different formalisms.

Multi-context systems (MCSs) provide a general framework for this kind of integration. The basic idea underlying MCSs is to leave the diverse formalisms and knowledge bases untouched, and to use so-called bridge rules to model the flow of information among different parts of the system. An MCS consists of reasoning units – called

[1] Institute of Computer Science, Leipzig University, Germany, email: {brewka|ellmauthaler|puehrer}@informatik.uni-leipzig.de.
[2] NOVA LINCS & Departamento de Informática, Universidade NOVA de Lisboa, Portugal, email: {rjrg|mkn|jleite}@fct.unl.pt.

contexts for historical reasons [22] – where each unit is equipped with a collection of bridge rules. In a nutshell, the bridge rules allow contexts to "listen" to other contexts, that is to take into account beliefs held in other contexts.

Bridge rules are similar to logic programming rules (including default negation), with an important difference: they provide means to access other contexts in their bodies. Bridge rules not only allow for a fully declarative specification of the information flow, but they also allow information to be modified instead of being just passed along as is. Using bridge rules we may translate a piece of information into the language/format of another context, pass on an abstraction of the original information, leaving out unnecessary details, select or hide information, add conclusions to a context based on the absence of information in another one, and even use simple encodings of preferences among parent contexts.

Historically, MCSs went through several development steps until they reached their present form. Advancing work in [21, 30] aiming to integrate different inference systems, monotonic heterogeneous multi-context systems were defined in [22], with reasoning within as well as across monotonic contexts. The first, still somewhat limited attempts to include non-monotonic reasoning were done in [33] and [10], where default negation in the rules is used to allow for reasoning based on the *absence* of information from a context.

The non-monotonic MCSs of [7] substantially generalize previous approaches, by accommodating *heterogeneous* and both *monotonic* and *non-monotonic* contexts, hence capable of integrating, among many others, "typical" monotonic logics like description logics or temporal logics, and non-monotonic formalisms like Reiter's default logic, logic programs under answer set semantics, circumscription, defeasible logic, or theories in autoepistemic logic. The semantics of nonmonotonic MCSs is defined in terms of equilibria: a belief set for each context that is acceptable for its knowledge base augmented by the heads of its applicable bridge rules.

More recently, the so-called managed MCSs (mMCSs) [8] addressed a limitation of MCSs in the way they integrate knowledge between contexts. Instead of simply *adding* the head of an applicable bridge rule to the context's knowledge base, which could cause some inconsistency, mMCSs allow for operations other than addition, such as, for instance, *revision* and *deletion*, hence dealing with the problem of consistency management within contexts.

Dynamics The second problem stems from the shift from static knowledge-based systems that 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.

Indeed, traditional knowledge-based systems – including the different variants of MCSs mentioned above – focus entirely on static situations, which is the right thing for applications such as for instance expert systems, configuration or planning problems, where the available background knowledge changes rather slowly, if at all, and where all that is needed is the solution of a new instance of a known problem. However, the new kinds of applications we consider are becoming more and more important, and these require continuous online reasoning, including observing and reacting to events.

There are some examples of systems developed with the purpose of reacting to streams of incoming information, such as Reactive ASP [19, 18], C-SPARQL [5], Ontology Streams [29] and ETALIS [2], to name only a few. However, they are very limited in the kind of knowledge that can be represented, and the kind of reasoning allowed, hence unsuitable to address the requirements of the applications we envision, such as those that need to integrate heterogeneous knowledge bases. Additionally, reacting to the streams of incoming information is only part of the dynamic requirements of our applications. In many cases, the incoming information is processed only once, perhaps requiring complex reasoning using various knowledge bases to infer the right way to react, and does not have to be dealt with again – e.g., concluding that nothing needs to be done after determining that the tachycardia is caused by the decongestant recently taken by Dave. In other cases, it is important that these observations not only influence the current reaction of the system – do nothing in the previous example – but, at the same time, be able to change the knowledge bases in a more permanent way, i.e., allowing for the internalization of knowledge. For example, relevant observations regarding Dave's health status should be added to his medical records, such as for example that he had an episode of tachycardia caused by a decongestant, and, in the future, maybe even revise such episode if it is found that Dave had forgotten to take the decongestant after all. Other more sophisticated changes in the knowledge bases include, for example, an update to the biomedical health ontology whenever new treatments are found, the revision of the policy rules whenever some exceptions are found, etc. EVOLP [1] extends logic programming under answer set semantics with the possibility to specify its evolution, through successive updates, in reaction to external observations. It is nevertheless limited to a single knowledge representation formalism and to a single operation (update).

In this paper, we aim to address these challenges. We develop a system that allows us to integrate heterogeneous knowledge bases with streams of incoming information and to use them for continuous online reasoning, reacting, and evolving the knowledge bases by internalizing relevant knowledge. To this end, we introduce *reactive Multi-Context Systems (rMCSs)*. These systems build upon mMCSs and thus provide their functionality for integrating heterogeneous knowledge sources, admitting also relevant operations on knowledge bases. In addition, rMCSs can handle continuous streams of input data. Equilibria remain the fundamental underlying semantic notion, but the focus now lies on the dynamic evolution of the systems. In a nutshell, given an initial configuration of knowledge bases, that is, an initial knowledge base for each context, a specific input stream will lead to a corresponding stream of equilibria, generated by respective updates of the knowledge bases. Contrary to standard MCSs which possess only one type of bridge rules modeling the information flow which needs to be taken into account when equilibria are computed (or the operations that need to be applied in case of mMCSs), rMCSs have an additional, different type of bridge rules, distinguished by the occurrence of the operator **next** in the head. These rules are used to specify how the configuration of knowledge bases evolves whenever an equilibrium was computed.

The *reactive Multi-Context Systems (rMCSs)* presented here combine and unify the two adaptations of multi-context systems for dynamic environments in [9] and [24], independently developed by different subsets of the authors. The approach presented here generalizes these earlier approaches and substantially improves on the presentation of the underlying concepts.

The occurrence of inconsistencies within frameworks that aim at integrating knowledge from different sources cannot be neglected, even more so in dynamic settings where knowledge changes over time. Even with the power of management operations that allow the specification of e.g. belief revision operations, many reasons remain why rMCSs may fail to have an equilibria stream, traceable to individual contexts, their interaction through the bridge rules, or their interaction with the input streams, which can render the entire system

useless. In this paper, we will address the problem of inexistent equilibria streams, also known as *global inconsistency*, following different strategies, such as repairing the rMCS, or even relaxing the notion of equilibria stream so that it can *go through* inconsistent states.

The paper is organized as follows. In Section 2, we introduce reactive MCSs, our framework for reactive reasoning in the presence of heterogeneous knowledge sources. In particular, we show how to integrate data streams into mMCSs and how to model the dynamics of our systems, based on two types of bridge rules. Reasoning based on multiple knowledge sources that need to be integrated faces the problem of potential inconsistencies. Section 3 discusses various methods for handling inconsistencies, with a special focus on non-existence of equilibria. We conclude and point out future directions of work in Section 4.

2 Reactive Multi-Context Systems

Reactive multi-context systems (rMCSs) make use of basic ideas from managed multi-context systems (mMCSs) [8] which extend multi-context systems (MCSs) as defined by Brewka and Eiter [7] by management capabilities. In particular, similar to mMCSs, we will make use of a management function and bridge rules that allow for conflict resolution between contexts as well as a fine-grained declarative specification of the information flow between contexts. To not unnecessarily burden the reader with repetetive material on these common components, we abstain from recalling the details of mMCSs first. It will be clear from the presentation when new concepts/ideas specific to rMCSs will be presented.

2.1 Specifying the Components of an rMCS

Similar as for previous notions of MCSs, we build on an abstract notion of a *logic*, which is a triple $L = \langle KB, BS, \mathbf{acc}\rangle$, where KB is the *set of admissible knowledge bases* of L, which are sets whose elements are called *knowledge base formulas*; BS is the *set of possible belief sets*, where elements in belief sets are called *beliefs*; and $\mathbf{acc} : KB \to 2^{BS}$ is a function describing the semantics of L by assigning to each knowledge base a set of *acceptable belief sets*.[3]

Example 1 *We illustrate how different formalisms can be represented by the notion of a logic. The logics presented below will serve as blueprints for the logics we use in examples throughout the paper. First, consider the case of classical propositional logic. Given a propositional signature Σ, we denote by F the set of all well-formed propositional formulas over Σ. To represent entailment in classical propositional logic over signature Σ, we consider the logic $L_p = \langle KB_p, BS_p, \mathbf{acc}_p\rangle$, such that the admissible knowledge bases are given by the set $KB_p = 2^F$. Since propositional logic aims at modeling ideal rational reasoning, we identify the set of possible belief sets BS_p with the set of deductively closed sets of formulas over Σ. Finally, $\mathbf{acc}_p$ maps every $kb \in KB_p$ to $\{E\}$, where E is the set of formulas entailed by kb.*

Quite similar in spirit are description logics (DLs) as (commonly decidable) fragments of first-order logic [4]. Given a DL language $\mathcal{L}$, we consider the logic $L_d = \langle KB_d, BS_d, \mathbf{acc}_d\rangle$ where KB_d is the set of all well-formed DL knowledge bases over $\mathcal{L}$, also called ontologies, BS_d is the set of deductively closed subsets of $\mathcal{L}$, and $\mathbf{acc}_d$ maps every $kb \in KB_d$ to $\{E\}$, where E is the set of formulas in $\mathcal{L}$ entailed by kb.

As an example for a non-deterministic formalism, consider logic programs under the answer set semantics [20]. Given a set of ground, i.e., variable-free, atoms A, we consider the logic $L_a = \langle KB_a, BS_a, \mathbf{acc}_a\rangle$, such that KB_a is the set of all logic programs over A. The set of possible belief sets is given by the set $BS_a = 2^A$ of possible answer sets and the function $\mathbf{acc}_a$ maps every logic program to the set of its answer sets.

Given a set E of entries, a simple logic for storing elements from E can be realized by the logic $L_s = \langle KB_s, BS_s, \mathbf{acc}_s\rangle$, such that $KB_s = BS_s = 2^E$, and $\mathbf{acc}_s$ maps every set $E' \subseteq E$ to $\{E'\}$. Such L_s can, e.g., be used to represent a simple database logic. We will call a logic of this type a storage logic.

In addition to a logic that captures language and semantics of a formalism to be integrated in an rMCS, a context also describes how a knowledge base belonging to the logic can be manipulated. To this end, we make use of a management function similar as in managed multi-context systems [8].

Definition 1 (Context) *A context is a triple $C = \langle L, OP, \mathbf{mng}\rangle$ where*

- *$L = \langle KB, BS, \mathbf{acc}\rangle$ is a logic,*
- *OP is a* set of operations*,*
- *$\mathbf{mng} : 2^{OP} \times KB \to KB$ is a* management function*.*

For an indexed context C_i we will write $L_i = \langle KB_i, BS_i, \mathbf{acc}_i\rangle$, OP_i, and $\mathbf{mng}_i$ to denote its components. Note that we leave the exact nature of the operations in OP unspecified. They can be seen as mere labels that determine how the management function should manipulate the knowledge base: we use subsets OP' of OP as the first argument of the management function that maps a knowledge base to an updated version depending on the presence or absence of operations in OP'. Thus, it is the management function that implements the semantics of the operations. We use a deterministic management function rather than a non-deterministic one as used in mMCS [8]. Note that it is straightforward to adapt our definitions to use non-deterministic management functions as well. However, as they are not essential to our approach, we here refrain from doing so to keep notation simpler.

Example 2 *Consider the assisted living scenario from the Introduction and remember that we target a system that recognizes potential threats caused by overheating of the stove in Dave's kitchen. We use the context C_{st} to monitor the stove. Its logic L_{st} is a storage logic taking $E = \{\mathrm{pw}, \mathrm{tm}(cold), \mathrm{tm}(hot)\}$ as the set of entries, representing the stove's power status (on if* pw *is present, and off otherwise) and a qualitative value for its temperature (cold/hot). At all times, the current temperature and power state of the stove should be stored in a knowledge base over L_{st}. Thus, the context provides the operations*

$$OP_{st} = \{\mathsf{setPower}(\mathit{off}), \mathsf{setPower}(\mathit{on}), \mathsf{setTemp}(\mathit{cold}), \mathsf{setTemp}(\mathit{hot})\}$$

to update the information in the knowledge base.

Before defining the management function we need to clarify what we want this function to achieve. We assume the existence of a single temperature sensor constantly providing a measurement which triggers exactly one of the setTemp *operations. For this reason, there is*

[3] To ease readability, throughout the paper, we will often use the following convention when writing symbols: single entities are lower-case, while sets of entities and structures with different components are upper-case; in addition, sequences of those are indicated in sans serif, while notions with a temporal dimension are written in calligraphic letters (only upper-case, such as $\mathcal{S}$ or $\mathcal{I}$); finally, operators and functions are **bold**.

no need for the management function to care about persistence of the temperature value, or about conflicting information. The power information, on the other hand, is based on someone toggling a switch, so we definitely need persistence of the fluent pw.

The semantics of the operations is thus given, for $OP' \subseteq OP_{st}$*, by the management function* $\mathbf{mng}_{st}(OP', kb) =$

$$\{\mathrm{pw} \mid \mathsf{setPower}(on) \in OP' \vee (\mathrm{pw} \in kb \wedge \mathsf{setPower}(off) \notin OP')\} \cup \{\mathrm{tm}(t) \mid \mathsf{setTemp}(t) \in OP'\}.$$

As discussed above, the function simply inserts the current qualitative temperature value. For the power, it ensures that the stove is considered on whenever it is switched on, and also when it is not being switched off and already considered on in the given knowledge base kb*. The second alternative implements persistence. Note that whenever both conflicting* setPower *operations are in* OP'*,* $\mathsf{setPower}(on)$ *"wins", that is,* pw *will be in the knowledge base. This is justified by the application: the damage of, say, unnecessarily turning off the electricity is a lot smaller than that of overlooking a potential overheating of the stove.*

Assume we have a knowledge base $kb = \{\mathrm{tm}(cold)\}$*. Then, an update with the set* $OP = \{\mathsf{setPower}(on), \mathsf{setTemp}(hot)\}$ *of operations would result in the knowledge base* $\mathbf{mng}_{st}(OP, kb) = \{\mathrm{pw}, \mathrm{tm}(hot)\}$.

Contexts exchange information by manipulating their associated knowledge bases using the management function. The central device for that is given by *bridge rules* that are rules similar in spirit to those in logic programming and that determine which operations from OP_i to apply to kb_i in a context C_i. In this work, we are interested in systems composed of contexts whose behavior may not only depend on other contexts, but also on input from the outside. Like communication between contexts, also external information is incorporated by means of bridge rules. To keep the approach as abstract as possible, all we require is that inputs are elements of some formal *input language* IL. Moreover, we allow for situations where input comes from different sources with potentially different input languages and thus consider tuples $\langle IL_1, \ldots, IL_k \rangle$ of input languages.

Definition 2 (Bridge Rule) *Let* $\mathsf{C} = \langle C_1, \ldots, C_n \rangle$ *be a tuple of contexts and* $\mathsf{IL} = \langle IL_1, \ldots, IL_k \rangle$ *a tuple of input languages. A* bridge rule for C_i over C and IL*,* $i \in \{1, \ldots, n\}$*, is of the form*

$$\mathbf{op} \leftarrow a_1, \ldots, a_j, \mathbf{not}\ a_{j+1}, \ldots, \mathbf{not}\ a_m \qquad (1)$$

such that $\mathbf{op} = op$ *or* $\mathbf{op} = \mathbf{next}(op)$ *for* $op \in OP_i$*,* $j \in \{0, \ldots, m\}$*, and every* atom a_ℓ*,* $\ell \in \{1, \ldots, m\}$*, is one of the following:*

- *a* context atom $c{:}b$ *with* $c \in \{1, \ldots, n\}$ *and* $b \in B$ *for some* $B \in BS_c$*, or*
- *an* input atom $s{::}b$ *with* $s \in \{1, \ldots, k\}$ *and* $b \in IL_s$.

For a bridge rule r *of the form (1)* $\mathrm{hd}(r)$ *denotes* $\mathbf{op}$*, the* head of r*, while* $\mathrm{bd}(r) = \{a_1, \ldots, a_j, \mathbf{not}\ a_{j+1}, \ldots, \mathbf{not}\ a_m\}$ *is the* body of r*. A* literal *is either an atom or the default negation of an atom, and we also differentiate between* context literals *and* input literals*.*

Roughly, a set of bridge rules for C_i describes which operations to apply to its knowledge base kb_i depending on whether currently available beliefs and external inputs match the literals in the body. We define and discuss their precise semantics later in Section 2.2. Bridge rules can be seen as the glue that binds the contexts in an rMCS together. Thus, we are now ready to define reactive multi-context systems.

Definition 3 (Reactive Multi-Context System) *A* reactive Multi-Context System (rMCS) *is a tuple* $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$*, where*

- $\mathsf{C} = \langle C_1, \ldots, C_n \rangle$ *is a tuple of contexts;*
- $\mathsf{IL} = \langle IL_1, \ldots, IL_k \rangle$ *is a tuple of input languages;*
- $\mathsf{BR} = \langle BR_1, \ldots, BR_n \rangle$ *is a tuple such that each* BR_i*,* $i \in \{1, \ldots, n\}$*, is a set of bridge rules for* C_i *over* C *and* IL.

Example 3 *We continue by using context* C_{st} *from Example 2 as the single context of the rMCS* $M_{ex3} = \langle \langle C_{st} \rangle, \langle IL_{ex3} \rangle, \langle BR_{ex3} \rangle \rangle$.[4] *The input language* $IL_{ex3} = \{\mathrm{switch}\}$ *is used to report whether the switch for changing the power state of the stove has been turned. The bridge rules in* BR_{ex3} *are given by*

$$\mathbf{next}(\mathsf{setPower}(on)) \leftarrow ex3{::}\mathrm{switch}, \mathbf{not}\ st{:}\mathrm{pw}.$$
$$\mathbf{next}(\mathsf{setPower}(off)) \leftarrow ex3{::}\mathrm{switch}, st{:}\mathrm{pw}.$$

and react to switching the stove on or off: depending on the current power state of the stove that is stored in a knowledge base associated to C_{st}*, whenever the switch is activated, the bridge rules derive an update of the knowledge base where the power state is reversed.*

2.2 Reacting to External Inputs - Semantics of rMCSs

To define the semantics of rMCSs, we first focus on the static case of a single time instant, and only subsequently introduce the corresponding dynamic notions for reacting to inputs changing over time.

We start with the evaluation of bridge rules, for which we need to know current beliefs and current external information. The former is captured by the notion of a *belief state* denoted by a tuple of belief sets – one for each context – similar as in previous work on multi-context systems.

Definition 4 (Belief State) *Let* $M = \langle \langle C_1, \ldots, C_n \rangle, \mathsf{IL}, \mathsf{BR} \rangle$ *be an rMCS. Then, a* belief state for M *is a tuple* $\mathsf{B} = \langle B_1, \ldots, B_n \rangle$ *such that* $B_i \in BS_i$*, for each* $i \in \{1, \ldots, n\}$*. We use* Bel_M *to denote the set of all beliefs states for* M.

In addition, to also capture the current external information, we introduce the notion of an *input*.

Definition 5 (Input) *Let* $M = \langle \mathsf{C}, \langle IL_1, \ldots, IL_k \rangle, \mathsf{BR} \rangle$ *be an rMCS. Then an* input for M *is a tuple* $\mathsf{I} = \langle I_1, \ldots, I_k \rangle$ *such that* $I_i \subseteq IL_i$*,* $i \in \{1, \ldots, k\}$*. The* set of all inputs for M *is denoted by* In_M.

We are now ready to define when literals (in bridge rule bodies) are satisfied.

Definition 6 (Satisfaction of Literals) *Let* $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$ *be an rMCS such that* $\mathsf{C} = \langle C_1, \ldots, C_n \rangle$ *and* $\mathsf{IL} = \langle IL_1, \ldots, IL_k \rangle$*. Given an input* $\mathsf{I} = \langle I_1, \ldots, I_k \rangle$ *for* M *and a belief state* $\mathsf{B} = \langle B_1, \ldots, B_n \rangle$ *for* M*, we define the satisfaction of literals given* I *and* B *as:*

- $\langle \mathsf{I}, \mathsf{B} \rangle \models a_\ell$ *if* a_ℓ *is of the form* $c{:}b$ *and* $b \in B_c$*;*
- $\langle \mathsf{I}, \mathsf{B} \rangle \models a_\ell$ *if* a_ℓ *is of the form* $s{::}b$ *and* $b \in I_s$*;*
- $\langle \mathsf{I}, \mathsf{B} \rangle \models \mathbf{not}\ a_\ell$ *if* $\langle \mathsf{I}, \mathsf{B} \rangle \not\models a_\ell$.

Let r *be a bridge rule for* C_i *over* C *and* IL*. Then*

[4] Throughout the paper we sometimes use labels (such as st in C_{st}) instead of numerical indices in our examples.

- $\langle \mathsf{I}, \mathsf{B} \rangle \models \mathrm{bd}(r)$ *if* $\langle \mathsf{I}, \mathsf{B} \rangle \models l$ *for every* $l \in \mathrm{bd}(r)$.

If $\langle \mathsf{I}, \mathsf{B} \rangle \models \mathrm{bd}(r)$, we say that r is *applicable under* I *and* B. The operations encoded in the heads of applicable bridge rules in an rMCS determine which knowledge base updates should take place. We collect them in two disjoint sets.

Definition 7 (Applicable Bridge Rules) *Let* $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$ *be an rMCS such that* $\mathsf{C} = \langle C_1, \ldots, C_n \rangle$ *and* $\mathsf{BR} = \langle BR_1, \ldots, BR_n \rangle$. *Given an input* I *for* M *and a belief state* B *for* M*, we define, for each* $i \in \{1, \ldots, n\}$*, the sets*

- $\mathbf{app}_i^{now}(\mathsf{I}, \mathsf{B}) = \{\mathrm{hd}(r) \mid r \in BR_i, \langle \mathsf{I}, \mathsf{B} \rangle \models \mathrm{bd}(r), \mathrm{hd}(r) \in OP_i\}$*;*
- $\mathbf{app}_i^{next}(\mathsf{I}, \mathsf{B}) = \{op \mid r \in BR_i, \langle \mathsf{I}, \mathsf{B} \rangle \models \mathrm{bd}(r), \mathrm{hd}(r) = \mathbf{next}(op)\}$.

Intuitively, the operations in $\mathbf{app}_i^{now}(\mathsf{I}, \mathsf{B})$ are used for computing temporary changes that influence the semantics of an rMCS for a single point in time. The operations in $\mathbf{app}_i^{next}(\mathsf{I}, \mathsf{B})$ on the other hand are used for changing knowledge bases over time. They are not used for computing the current semantics but are applied in the next point in time depending on the current semantics. This continuous change of knowledge bases over time is the reason why, unlike in previous work on MCSs, we do not consider knowledge bases as part of the contexts to which they are associated but store them in a separate configuration structure defined next.

Definition 8 (Configuration of Knowledge Bases) *Let* $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$ *be an rMCS such that* $\mathsf{C} = \langle C_1, \ldots, C_n \rangle$*. A* configuration of knowledge bases for M *is a tuple* $\mathsf{KB} = \langle kb_1, \ldots, kb_n \rangle$ *such that* $kb_i \in KB_i$*, for each* $i \in \{1, \ldots, n\}$*. We use* Con_M *to denote the set of all configurations of knowledge bases for* M.

The semantics of an rMCS for a single time instant is given in terms of its *equilibria*.

Definition 9 (Equilibrium) *Let* $M = \langle \langle C_1, \ldots, C_n \rangle, \mathsf{IL}, \mathsf{BR} \rangle$ *be an rMCS,* $\mathsf{KB} = \langle kb_1, \ldots, kb_n \rangle$ *a configuration of knowledge bases for* M*, and* I *an input for* M*. Then, a belief state* $\mathsf{B} = \langle B_1, \ldots, B_n \rangle$ *for* M *is an* equilibrium *of* M *given* KB *and* I *if, for each* $i \in \{1, \ldots, n\}$*, we have that*

$$B_i \in \mathbf{acc}_i(kb'), \text{ where } kb' = \mathbf{mng}_i(\mathbf{app}_i^{now}(\mathsf{I}, \mathsf{B}), kb_i).$$

Example 4 *Consider rMCS* M_{ex3} *from Example 3 and the configuration* $\mathsf{KB} = \langle kb_{st} \rangle$ *of knowledge bases for* M_{ex3} *with* $kb_{st} = \emptyset$*, representing that the stove is turned off. Moreover, consider the input* $\mathsf{I} = \langle \{\mathrm{switch}\} \rangle$ *for* M_{ex3} *and the belief state* $\mathsf{B} = \langle \emptyset \rangle$ *for* M_{ex3}*. As both bridge rules in* BR_{ex3} *use the* $\mathbf{next}$ *operator, we have* $\mathbf{app}_{st}^{now}(\mathsf{I}, \mathsf{B}) = \emptyset$ *and consequently, following the definition of the management function* $\mathbf{mng}_{st}$ *in Example 2,* kb_{st} *remains unchanged, i.e.,* $\mathbf{mng}_{st}(\mathbf{app}_{st}^{now}(\mathsf{I}, \mathsf{B}), kb_{st}) = kb_{st}$*. Thus,* $\mathbf{acc}_{st}(\mathbf{mng}_{st}(\mathbf{app}_{st}^{now}(\mathsf{I}, \mathsf{B}), kb_{st})) = \{\emptyset\}$*. It follows that* B *is an equilibrium of* M_{ex3} *given* KB *and* I.

Based on an equilibrium at the current time instant, we can compute an updated configuration of knowledge bases using the update function as introduced in the following.

Definition 10 (Update Function) *Let* $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$ *be an rMCS such that* $\mathsf{C} = \langle C_1, \ldots, C_n \rangle$*,* $\mathsf{KB} = \langle kb_1, \ldots, kb_n \rangle$ *a configuration of knowledge bases for* M*,* I *an input for* M*, and* B *a belief state for* M*. Then, the* update function for M *is defined as* $\mathbf{upd}_M(\mathsf{KB}, \mathsf{I}, \mathsf{B}) = \langle kb'_1, \ldots, kb'_n \rangle$*, such that, for each* $i \in \{1 \ldots, n\}$*,* $kb'_i = \mathbf{mng}_i(\mathbf{app}_i^{next}(\mathsf{I}, \mathsf{B}), kb_i)$.

With all this in place, we can finally show how an rMCS behaves in the presence of external information that changes over time. For this purpose, we assume that an rMCS receives data in a stream of inputs, i.e., an input for each time instant, and we represent individual time instants by natural numbers. These can be interpreted as logical time instants that do not necessarily represent specific physical time points. In particular, we do not assume that every pair of consecutive natural numbers represents equidistant physical time spans.

Definition 11 (Input Stream) *Let* $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$ *be an rMCS such that* $\mathsf{IL} = \langle IL_1, \ldots, IL_k \rangle$*. An* input stream for M *(until* τ*) is a function* $\mathcal{I} : [1..\tau] \to \mathsf{In}_M$ *where* $\tau \in \mathbb{N} \cup \{\infty\}$.

We will conveniently omit the term "until τ" whenever the limit of the stream is irrelevant. Conversely, we will provide an explicit τ, when we are only interested in the first τ time instants or when we want to state that the input stream is infinite (in case $\tau = \infty$). Clearly, an input stream for M until τ also fully determines an input stream for M until τ' for every $1 \leq \tau' < \tau$.

For any stream, we commonly represent the functional notation by a superscript, e.g., given an input stream $\mathcal{I}$ and $t \in [1..\tau]$, we will use $\mathcal{I}^t$ to denote $\mathcal{I}(t)$, i.e., the input $\langle I_1, \ldots, I_k \rangle$ for M at time t. We also term *stream* the restriction of an input stream $\mathcal{I}$ to a single input language IL_i, which can be understood as a function $\mathcal{I}_i : [1..\tau] \to 2^{IL_i}$ that is fully determined by $\mathcal{I}$.

Note that $\mathcal{I}^t$ encapsulates (input) data for every input language of M. Hence, we assume that, at every time instant, we have information from every external source of the rMCS. This synchronous approach is required since the evaluation of a bridge rule may depend on the availability of information from multiple streams. One possibility for modeling external sources that do not continuously provide information is setting $\mathcal{I}_s^t$ to the empty set for representing a lack of input from the source with language IL_s at time t.

The semantics of an rMCS over time is given by its *equilibria streams* for a given initial configuration of knowledge bases and an input stream for the system.

Definition 12 (Equilibria Stream) *Let* $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$ *be an rMCS,* KB *a configuration of knowledge bases for* M*, and* $\mathcal{I}$ *an input stream for* M *until* τ *where* $\tau \in \mathbb{N} \cup \{\infty\}$*. Then, an* equilibria stream of M given KB and $\mathcal{I}$ *is a function* $\mathcal{B} : [1..\tau] \to \mathsf{Bel}_M$ *such that*

- $\mathcal{B}^t$ *is an equilibrium of* M *given* $\mathcal{KB}^t$ *and* $\mathcal{I}^t$*, where* $\mathcal{KB}^t$ *is inductively defined as*
  - $\mathcal{KB}^1 = \mathsf{KB}$
  - $\mathcal{KB}^{t+1} = \mathbf{upd}_M(\mathcal{KB}^t, \mathcal{I}^t, \mathcal{B}^t)$.

We will also refer to the function $\mathcal{KB} : [1..\tau] \to \mathsf{Con}_M$ *as the* configurations stream of M given KB, $\mathcal{I}$, and $\mathcal{B}$.

Note that the limit τ of an equilibria stream is aligned with that of the given input stream. Following the definition, it is easy to see that if we have an equilibria stream $\mathcal{B}$ of M given KB and $\mathcal{I}$, then the substream of $\mathcal{B}$ of size τ', with $\tau' \leq \tau$, is an equilibria stream of M given KB and $\mathcal{I}'$, where $\mathcal{I}'$ is the substream of $\mathcal{I}$ of size τ'. This implies that, conversely, each extension of the input stream can only lead to equilibria streams that extend those obtained given the original input stream.

Example 5 *Reconsider rMCS* M_{ex3}*,* KB*, and* B *from Example 4, as well as an input stream* $\mathcal{I}$ *until* 3 *with* $\mathcal{I}^1 = \langle \{\mathrm{switch}\} \rangle$*,* $\mathcal{I}^2 = \langle \emptyset \rangle$*,*

and $\mathcal{I}^3 = \langle\{\text{switch}\}\rangle$. *There is an equilibria stream* $\mathcal{B}$ *of* M_{ex3} *given* KB *and* $\mathcal{I}$. *Note that the input* $\mathcal{I}^1$ *coincides with input* I *from Example 4. As* B *is the only equilibrium of* M_{ex3} *given* KB *and* I, *we have that* $\mathcal{B}^1 = \mathsf{B}$.

As we have $\mathbf{app}^{next}_{st}(\mathsf{I}, \mathsf{B}) = \{\mathsf{setPower}(on)\}$, *the update function provides the following configuration of knowledge bases for time instant* 2 *(with* $\mathcal{KB}^1 = \mathsf{KB}$*):*

$$\mathcal{KB}^2 = \mathbf{upd}_{M_{ex3}}(\mathcal{KB}^1, \mathcal{I}^1, \mathcal{B}^1) = \\ = \langle \mathbf{mng}_{st}(\mathbf{app}^{next}_{st}(\mathsf{I}, \mathsf{B}), kb)\rangle = \langle\{\text{pw}\}\rangle.$$

Thus, switching the power state at time 1 *leads to an updated knowledge base indicating that the stove is on at time* 2. *The table in Fig. 1 summarizes the equilibria stream and the configurations stream given* KB *and* $\mathcal{I}$. *Note that, due to the choice of logic and since all bridge rules use the next operator, equilibria necessarily coincide with the corresponding knowledge base at each time step.*

t	$\mathcal{KB}^t$	$\mathcal{I}^t$	$\mathcal{B}^t$	$\mathbf{app}^{next}_{st}(\mathcal{I}^t, \mathcal{B}^t)$
1	$\langle\emptyset\rangle$	$\langle\{\text{switch}\}\rangle$	$\langle\emptyset\rangle$	$\{\mathsf{setPower}(on)\}$
2	$\langle\{\text{pw}\}\rangle$	$\langle\emptyset\rangle$	$\langle\{\text{pw}\}\rangle$	$\emptyset$
3	$\langle\{\text{pw}\}\rangle$	$\langle\{\text{switch}\}\rangle$	$\langle\{\text{pw}\}\rangle$	$\{\mathsf{setPower}(off)\}$

Figure 1: Streams and applicable operations for M_{ex3}

3 Inconsistency Management

The occurrence of inconsistencies within frameworks that aim at integrating knowledge from different sources cannot be neglected, even more so in dynamic settings where knowledge changes over time. There are many reasons why rMCSs may fail to have an equilibria stream. These include the absence of an acceptable belief set for one of its contexts given its current knowledge base at some point in time, some occurring conflict between the operations in the heads of bridge rules, or simply because the input stream is such that the configuration of the flow of information within the rMCS, namely its bridge rules, prevent the existence of such an equilibria stream. In a real world situation, an rMCS without an equilibria stream is essentially useless. Not only can it not be used at the first time point equilibria ceased to exist, but it also cannot recover, even if what caused the problem was the particular input at that time point, which is bound to subsequently change into some other input that would no longer cause any trouble. This is so because an equilibria stream requires the existence of an equilibrium at every time point.

In this section, we address the problem of inexistent equilibria streams, also known as *global inconsistency*. We begin by defining a notion of coherence associated with individual contexts which allows us to first establish sufficient conditions for the existence of equilibria streams, and then abstract away from problems due to specific incoherent contexts and focus on those problems essentially caused by the way the flow of information in rMCSs is organized through its bridge rules. We introduce the notion of a *repair*, which modifies an rMCS by changing its bridge rules at some particular point in time in order to obtain some equilibria stream, which we dub *repaired equilibria stream*. We establish sufficient conditions for the existence of repaired equilibria streams and briefly discuss different possible strategies to define such repairs. However, repaired equilibria streams may not always exist either, e.g., because some particular context is incoherent. To deal with such situations, we relax the concept of equilibria stream and introduce the notion of *partial equilibria stream*, which essentially allows the non-existences of equilibria at some time points. It turns out that *partial equilibria streams* always exist thus solving the problem of global inconsistency for rMCSs.

In [12] the authors addressed the problem of global inconsistency in the context of mMCSs. Just as we do here, they begin by establishing sufficient conditions for the existence of equilibria. Then, they define the notions of *diagnosis* and *explanation*, the former corresponding to rules that need to be altered to restore consistency, and the latter corresponding to combinations of rules that cause inconsistency. These two notions turn out to be dual of each other, and somehow correspond to our notion of repair, the main difference being that, unlike in [12], we opt not to allow the (non-standard) strengthening of bridge-rule to restore consistency, and, of course, that fact that our repairs need to take into account the dynamic nature of rMCSs.

We start by introducing two notions of global consistency differing only on whether we consider a particular input stream or all possible input streams.

Definition 13 *Let* M *be an rMCS,* KB *a configuration of knowledge bases for* M, *and* $\mathcal{I}$ *an input stream for* M. *Then,* M *is* consistent *with respect to* KB *and* $\mathcal{I}$ *if there exists an equilibria stream of* M *given* KB *and* $\mathcal{I}$. M *is* strongly consistent *with respect to* KB *if, for every input stream* $\mathcal{I}$ *for* M, M *is consistent with respect to* KB *and* $\mathcal{I}$.

Obviously, for a fixed configuration of knowledge bases, strong consistency implies consistency w.r.t. any input stream, but not vice-versa.

Unfortunately, verifying strong consistency is in general highly complex since it requires checking all possible equilibria streams. Nevertheless, we can establish conditions that ensure that an rMCS M is strongly consistent with respect to a given configuration of knowledge bases KB, hence guaranteeing the existence of an equilibria stream independently of the input. It is based on two notions – *totally coherent contexts* and *acyclic rMCSs* – that together are sufficient to ensure (strong) consistency.

Total coherence imposes that each knowledge base of a context always has at least one acceptable belief set.

Definition 14 *A context* C_i *is* totally coherent *if* $\mathbf{acc}_i(kb) \neq \emptyset$, *for every* $kb \in KB_i$.

The second notion describes cycles between contexts which may be a cause of inconsistency. Acyclic rMCSs are those whose bridge rules have no cycles.

Definition 15 *Given an rMCS* $M = \langle\langle C_1, \ldots, C_n\rangle, \mathsf{IL}, \mathsf{BR}\rangle$, $\lhd_M$ *is the binary relation over contexts of* M *such that* $(C_i, C_j) \in \lhd_M$ *if there is a bridge rule* $r \in BR_i$ *and* $j{:}b \in \text{bd}(r)$ *for some* b. *If* $(C_i, C_j) \in \lhd_M$, *also denoted by* $C_i \lhd_M C_j$, *we say that* C_i *depends on* C_j *in* M, *dropping the reference to* M *whenever unambiguous.*

Definition 16 *An rMCS* M *is* acyclic *if the transitive closure of* $\lhd_M$ *is irreflexive.*

We can show that these two conditions together are indeed sufficient to ensure strong consistency.

Proposition 1 *Let* $M = \langle\langle C_1, \ldots, C_n\rangle, \mathsf{IL}, \mathsf{BR}\rangle$ *be an acyclic rMCS such that every* C_i, $1 \leq i \leq n$, *is totally coherent, and* KB *a configuration of knowledge bases for* M. *Then,* M *is strongly consistent with respect to* KB.

A similar property holds for consistent mMCSs, which indicates that the extension to rMCSs as such does not decrease the likelihood of existence of an equilibria stream. Nevertheless, these conditions are rather restrictive since there are many useful cyclic rMCSs which only under some particular configurations of knowledge bases and input streams may have no equilibria streams.

To deal with these, and recover an equilibria stream, one possibility is to repair the rMCSs by locally, and selectively, eliminating some of its bridge rules. Towards introducing the notion of *repair*, given an rMCS $M = \langle\langle C_1, \ldots, C_n\rangle, \mathsf{IL}, \mathsf{BR}\rangle$, we denote by br_M the set of all bridge rules of M, i.e., $br_M = \bigcup_{1\leq i\leq n} BR_i$. Moreover, given a set $R \subseteq br_M$, denote by $M[R]$ the rMCS obtained from M by restricting the bridge rules to those not in R.

Definition 17 (Repair) *Let $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR}\rangle$ be an rMCS, KB a configuration of knowledge bases for M, and $\mathcal{I}$ an input stream for M until τ where $\tau \in \mathbb{N} \cup \{\infty\}$. Then, a repair for M given KB and $\mathcal{I}$ is a function $\mathcal{R} : [1..\tau] \to 2^{br_M}$ such that there exists a function $\mathcal{B} : [1..\tau] \to \mathsf{Bel}_M$ such that*

- *$\mathcal{B}^t$ is an equilibrium of $M[\mathcal{R}^t]$ given $\mathcal{KB}^t$ and $\mathcal{I}^t$, where $\mathcal{KB}^t$ is inductively defined as*
  - *$\mathcal{KB}^1 = \mathsf{KB}$*
  - *$\mathcal{KB}^{t+1} = \mathbf{upd}_{M[\mathcal{R}^t]}(\mathcal{KB}^t, \mathcal{I}^t, \mathcal{B}^t)$.*

We refer to $\mathcal{B}$ as a repaired equilibria stream of M given KB, $\mathcal{I}$ and $\mathcal{R}$.

Note the generality of this notion, which considers to be a *repair* essentially any sequence of bridge rules (defined by the repair function $\mathcal{R}$) that, if removed from the rMCS at their corresponding time point, will allow for an equilibrium at that time point. This may include repairs that unnecessarily eliminate some bridge rules, and even the *empty repair* i.e. the repair $\mathcal{R}_\emptyset$ such that $\mathcal{R}^t_\emptyset = \emptyset$ for every t, whenever M already has an equilibria stream given KB and $\mathcal{I}$. This ensures that the set of repaired equilibria streams properly extends the set of equilibria streams, since equilibria streams coincide with repaired equilibria streams given the empty repair.

Proposition 2 *Every equilibria stream of M given KB and $\mathcal{I}$ is a repaired equilibria stream of M given KB, $\mathcal{I}$ and the empty repair $\mathcal{R}_\emptyset$.*

It turns out that for rMCSs composed of totally coherent contexts, repaired equilibria streams always exist.

Proposition 3 *Let $M = \langle\langle C_1, \ldots, C_n\rangle, \mathsf{IL}, \mathsf{BR}\rangle$ be an rMCS such that each C_i, $i \in \{1, \ldots, n\}$, is totally coherent, KB a configuration of knowledge bases for M, and $\mathcal{I}$ an input stream for M until τ. Then, there exists $\mathcal{R} : [1..\tau] \to 2^{br_M}$ and $\mathcal{B} : [1..\tau] \to \mathsf{Bel}_M$ such that $\mathcal{B}$ is a repaired equilibria stream given KB, $\mathcal{I}$ and $\mathcal{R}$.*

Whenever repair operations are considered in the literature, e.g., in the context of databases [3], there is a special emphasis on seeking repairs that are somehow minimal, the rational being that we want to change things as little as possible to regain consistency. In the case of repairs of rMCS, it is easy to establish an order relation between them, based on a comparison of the bridge rules to be deleted at each time point.

Definition 18 *Let $\mathcal{R}_a$ and $\mathcal{R}_b$ be two repairs for some rMCS M given a configuration of knowledge bases for M, KB and $\mathcal{I}$, an input stream for M until τ. We say that $\mathcal{R}_a \leq \mathcal{R}_b$ if $\mathcal{R}^i_a \subseteq \mathcal{R}^i_b$ for every $i \leq \tau$, and that $\mathcal{R}_a < \mathcal{R}_b$ if $\mathcal{R}_a \leq \mathcal{R}_b$ and $\mathcal{R}^i_a \subset \mathcal{R}^i_b$ for some $i \leq \tau$.*

This relation can be straightforwardly used to check whether a repair is minimal, and we can restrict ourselves to adopting minimal repairs. However, there may be good reasons to adopt non-minimal repairs, e.g., so that they can be determined *as we go*, or so that *deleted* bridge rules are not reinstated, etc. Even though investigating specific types of repairs falls outside the scope of this paper, we nevertheless present and briefly discuss some possibilities.

Definition 19 (Types of Repairs) *Let $\mathcal{R}$ be a repair for some rMCS M given KB and $\mathcal{I}$. We say that $\mathcal{R}$ is a:*

Minimal Repair *if there is no repair $\mathcal{R}_a$ for M given KB and $\mathcal{I}$ such that $\mathcal{R}_a < \mathcal{R}$.*
Global Repair *if $\mathcal{R}^i = \mathcal{R}^j$ for every $i, j \leq \tau$.*
Minimal Global Repair *if $\mathcal{R}$ is global and there is no global repair $\mathcal{R}_a$ for M given KB and $\mathcal{I}$ such that $\mathcal{R}_a < \mathcal{R}$.*
Incremental Repair *if $\mathcal{R}^i \subseteq \mathcal{R}^j$ for every $i \leq j \leq \tau$.*
Minimally Incremental Repair *if $\mathcal{R}$ is incremental and there is no incremental repair $\mathcal{R}_a$ and $j \leq \tau$ such that $\mathcal{R}^i_a \subset \mathcal{R}^i$ for every $i \leq j$.*

Minimal repairs perhaps correspond to the ideal situation in the sense that they never unnecessarily remove bridge rules. In some circumstances, it may be the case that if a bridge rule is somehow involved in some inconsistency, it should not be used at any time point, leading to the notion of *global repair*. Given the set of all repairs, checking which are global is also obviously less complex than checking which are minimal. A further refinement – *minimal global repairs* – would be to only consider repairs that are minimal among the global ones, which would be much simpler to check than checking whether it is simply minimal. Note that a minimal global repair is not necessarily a minimal repair. One of the problems with these types of repairs is that we can only globally check whether they are of that type, i.e., we can only check once we know the entire input stream $\mathcal{I}$. This was not the case with *plain* repairs, as defined in Def. 17, which could be checked *as we go*, i.e., we can determine what bridge rules to include in the repair at a particular time point by having access to the input stream $\mathcal{I}$ up to that time point only. This is important so that rMCSs can be used to effectively react to their environment. The last two types of repairs defined above allow for just that. *Incremental repairs* essentially impose that removed bridge rules cannot be reused in the future, i.e., that the set of removed bridge rules monotonically grows with time, while *minimally incremental repairs* further impose that only minimal sets of bridge rules can be added at each time point. Other types of repairs could be defined, e.g., by defining some priority relation between bridge rules, some distance measure between subsets of bridge rules and minimize it when considering the repair at consecutive time points, among many other options, whose investigation we leave for future work. Repairs could also be extended to allow for the strengthening of bridge rules, besides their elimination, generalizing ideas from [12] and [8] where the extreme case of eliminating the entire body of bridge rules as part of a repair is considered.

Despite the existence of repaired equilibria streams for large classes of systems, two problems remain: first, computing a repair may be excessively complex, and second, there remain situations where no repaired equilibria stream exists, namely when the rMCS contains contexts that are not totally coherent. The second issue could be dealt with by ensuring that for each non-totally coherent context there would be some bridge rule with a management operation in its head that would always restore consistency of the context, and that such rule could always be *activated* through a repair (for

example, by adding a negated reserved atom to its body, and another bridge rule with that atom in its head and an empty body, so that removing this latter rule through a repair would activate the management function and restore consistency of the context). But this would require special care in the way the system is specified, and its analysis would require a very complex analysis of the entire system including the specific behavior of management functions. In practice, it would be quite hard – close to impossible in general – to ensure the existence of repaired equilibria streams, and we would still be faced with the first problem, that of the complexity of determining the repairs.

A solution to this problem is to relax the notion of equilibria stream so that it does not require an equilibrium at every time point. This way, if no equilibrium exists at some time point, the equilibria stream would be undefined at that point, but possibly defined again in subsequent time points. This leads to the following notion of *partial equilibria stream*.

Definition 20 (Partial Equilibria Stream) *Let $M = \langle \mathsf{C}, \mathsf{IL}, \mathsf{BR} \rangle$ be an rMCS, KB a configuration of knowledge bases for M, and $\mathcal{I}$ an input stream for M until τ where $\tau \in \mathbb{N} \cup \{\infty\}$. Then, a* partial equilibria stream of M given KB and $\mathcal{I}$ *is a partial function $\mathcal{B} : [1..\tau] \nrightarrow \mathsf{Bel}_M$ such that*

- *$\mathcal{B}^t$ is an equilibrium of M given $\mathcal{KB}^t$ and $\mathcal{I}^t$, where $\mathcal{KB}^t$ is inductively defined as*
  - $\mathcal{KB}^1 = \mathsf{KB}$
  - $\mathcal{KB}^{t+1} = \begin{cases} \mathbf{upd}_M(\mathcal{KB}^t, \mathcal{I}^t, \mathcal{B}^t), & \text{if } \mathcal{B}^t \text{ is not undefined.} \\ \mathcal{KB}^t, & \text{otherwise.} \end{cases}$
- *or $\mathcal{B}^t$ is undefined.*

As expected, this is a proper generalisation of the notion of equilibria stream:

Proposition 4 *Every equilibria stream of M given KB and $\mathcal{I}$ is a partial equilibria stream of M given KB and $\mathcal{I}$*

And it turns out that partial equilibria streams always exist.

Proposition 5 *Let M be an rMCS, KB a configuration of knowledge bases for M, and $\mathcal{I}$ an input stream for M until τ. Then, there exists $\mathcal{B} : [1..\tau] \nrightarrow \mathsf{Bel}_M$ such that $\mathcal{B}$ is a partial equilibria stream given KB and $\mathcal{I}$.*

One final word to note is that partial equilibria streams not only allow us to deal with situations where equilibria do not exist at some time instants, but they also open the ground to consider other kinds of situations where we do not wish to consider equilibria at some time point, for example because we were not able to compute them on time, or simply because we do not wish to process the input at every time point, e.g., whenever we just wish to sample the input with a lower frequency than it is generated. If we wish to restrict that partial equilibria streams only relax equilibria streams when necessary, i.e., when equilibria do not exist at some time point, we can further impose the following condition on Def. 20:

$\mathcal{B}^t$ is undefined $\Rightarrow$ there is no equilibrium of M given $\mathcal{KB}^t$ and $\mathcal{I}^t$.

4 Conclusions

In this paper, we introduced *reactive Multi-Context Systems (rMCSs)*, a combination and unification of the two adaptations of multi-context systems for dynamic environments in [9] and [24], which generalizes and substantially improves on the presentation of the underlying concepts of these earlier approaches. Building upon mMCSs, rMCSs inherit their functionality for integrating heterogeneous knowledge sources, admitting also relevant operations on knowledge bases. In addition, rMCSs can handle continuous streams of input data. Equilibria remain the fundamental underlying semantic notion, but the focus now lies on the dynamic evolution of the systems.

Since we cannot ignore the possibility that inconsistencies may occur, which result in the absence of equilibria at certain time points ultimately rendering the entire system useless, we addressed this problem first by showing sufficient conditions on the contexts and the bridge rules that ensure the existence of an equilibria stream. In the cases where these conditions are not met, we presented two possible solutions, one following an approach based on repairs – essentially the selective removal of bridge rules to regain an equilibria stream – and a second by relaxing the notion of equilibria stream to ensure that intermediate inconsistent states can be recovered from.

There is much more to be done with respect to rMCSs, some of which we are already working on.

Nondeterminism is inherent to rMCSs due to, e.g., the flow of information between contexts established by bridge rules. This can be overcome by introducing preferences on equilibria using, e.g., preference functions as proposed in [14], or simply by preferring equilibria that represent minimal change between states, as proposed in [25]. One might also adopt language constructs for expressing preferences in ASP such as optimization statements [17] or weak constraints [11], which essentially assign a quality measure to equilibria. One way of avoiding nondeterminism is by applying an alternative, skeptical semantics – the well-founded semantics – along the lines of what was proposed in [28].

An alternative to deal with inconsistent states is to follow a paraconsistent approach, as proposed for hybrid knowledge bases in [16, 26]. Also, just as with EVOLP [1] and explored in [23], we could allow the bridge rules to change with time, strengthening the evolving and adaptation capabilities of rMCSs.

Finally, we need to compare our system with related work, including two of the most relevant approaches w.r.t. stream reasoning, namely LARS (Logic-based framework for Analyzing Reasoning over Streams) [6] and STARQL [32]. Additionally, in [15] the authors defined Asynchronous MCSs (aMCSs) a framework for loosely coupling knowledge representation formalisms. Contrarily to rMCSs, the semantics is not defined in terms of equilibria but instead an asynchronous semantics is defined assuming that every context delivers output whenever available. Establishing bridges between both systems would also shed new light into the synchronization problems associated with combining dynamic knowledge bases.

ACKNOWLEDGEMENTS

We would like to thank the reviewers for their comments, which helped improve this paper. R. Gonçalves, M. Knorr and J. Leite were partially supported by FCT under strategic project NOVA LINCS (PEst/UID/CEC/04516/2013). R. Gonçalves was partially supported by FCT grant SFRH/BPD/100906/2014 and M. Knorr by FCT grant SFRH/BPD/86970/2012. Moreover, G. Brewka, S. Ellmauthaler, and J. Pührer were partially supported by the German Research Foundation (DFG) under grants BR-1817/7-1 and FOR 1513.

References

[1] José Júlio Alferes, Antonio Brogi, João Alexandre Leite, and Luís Moniz Pereira, 'Evolving logic programs', in *Procs. of JELIA*, eds., Sergio Flesca, Sergio Greco, Nicola Leone, and Giovambattista Ianni, volume 2424 of *LNCS*, pp. 50–61. Springer, (2002).

[2] D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic, 'Stream reasoning and complex event processing in ETALIS', *Semantic Web*, **3**(4), 397–407, (2012).

[3] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki, 'Consistent query answers in inconsistent databases', in *Procs. of PODS*, eds., Victor Vianu and Christos H. Papadimitriou, pp. 68–79. ACM Press, (1999).

[4] *The Description Logic Handbook: Theory, Implementation, and Applications*, eds., Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, Cambridge University Press, 2nd edn., 2007.

[5] D. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus, 'C-SPARQL: a continuous query language for RDF data streams', *Int. J. Semantic Computing*, **4**(1), 3–25, (2010).

[6] Harald Beck, Minh Dao-Tran, Thomas Eiter, and Michael Fink, 'LARS: A logic-based framework for analyzing reasoning over streams', in *Procs. of AAAI*, eds., Blai Bonet and Sven Koenig, pp. 1431–1438. AAAI Press, (2015).

[7] Gerhard Brewka and Thomas Eiter, 'Equilibria in heterogeneous nonmonotonic multi-context systems', in *Procs. of AAAI*, pp. 385–390. AAAI Press, (2007).

[8] Gerhard Brewka, Thomas Eiter, Michael Fink, and Antonius Weinzierl, 'Managed multi-context systems', in *Procs. of IJCAI*, ed., Toby Walsh, pp. 786–791. IJCAI/AAAI, (2011).

[9] Gerhard Brewka, Stefan Ellmauthaler, and Jörg Pührer, 'Multi-context systems for reactive reasoning in dynamic environments', in *Procs. of ECAI*, pp. 159–164, (2014).

[10] Gerhard Brewka, Floris Roelofsen, and Luciano Serafini, 'Contextual default reasoning', in *Procs. of IJCAI*, ed., Manuela M. Veloso, pp. 268–273, (2007).

[11] Francesco Buccafurri, Nicola Leone, and Pasquale Rullo, 'Strong and weak constraints in disjunctive datalog', in *Procs. of LPNMR*, eds., Jürgen Dix, Ulrich Furbach, and Anil Nerode, volume 1265 of *LNCS*, pp. 2–17. Springer, (1997).

[12] Thomas Eiter, Michael Fink, Peter Schüller, and Antonius Weinzierl, 'Finding explanations of inconsistency in multi-context systems', *Artif. Intell.*, **216**, 233–274, (2014).

[13] Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits, 'Combining answer set programming with description logics for the semantic web', *Artif. Intell.*, **172**(12-13), 1495–1539, (2008).

[14] Stefan Ellmauthaler, 'Generalizing multi-context systems for reactive stream reasoning applications', in *Procs. of ICCSW*, eds., Andrew V. Jones and Nicholas Ng, volume 35 of *OASICS*, pp. 19–26. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, (2013).

[15] Stefan Ellmauthaler and Jörg Pührer, 'Asynchronous multi-context systems', in *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation - Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday*, eds., Thomas Eiter, Hannes Strass, Miroslaw Truszczynski, and Stefan Woltran, volume 9060 of *LNCS*, pp. 141–156. Springer, (2015).

[16] Michael Fink, 'Paraconsistent hybrid theories', in *Procs. of KR*, eds., Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith. AAAI Press, (2012).

[17] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele, *A users guide to gringo, clasp, clingo, and iclingo*, Potassco Team, 2010.

[18] Martin Gebser, Torsten Grote, Roland Kaminski, Philipp Obermeier, Orkunt Sabuncu, and Torsten Schaub, 'Stream reasoning with answer set programming: Preliminary report', in *Procs. of KR*, eds., Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, pp. 613–617. AAAI Press, (2012).

[19] Martin Gebser, Torsten Grote, Roland Kaminski, and Torsten Schaub, 'Reactive answer set programming', in *Procs. of LPNMR*, eds., James P. Delgrande and Wolfgang Faber, volume 6645 of *LNCS*, pp. 54–66. Springer, (2011).

[20] Michael Gelfond and Vladimir Lifschitz, 'Classical negation in logic programs and disjunctive databases', *New Generation Comput.*, **9**(3/4), 365–386, (1991).

[21] Fausto Giunchiglia, 'Contextual reasoning', *Epistemologia*, **XVI**, 345–364, (1993).

[22] Fausto Giunchiglia and Luciano Serafini, 'Multilanguage hierarchical logics or: How we can do without modal logics', *Artif. Intell.*, **65**(1), 29–70, (1994).

[23] Ricardo Gonçalves, Matthias Knorr, and João Leite, 'Evolving bridge rules in evolving multi-context systems', in *Procs. of CLIMA*, eds., Nils Bulling, Leendert W. N. van der Torre, Serena Villata, Wojtek Jamroga, and Wamberto Weber Vasconcelos, volume 8624 of *LNCS*, pp. 52–69. Springer, (2014).

[24] Ricardo Gonçalves, Matthias Knorr, and João Leite, 'Evolving multi-context systems', in *Procs. of ECAI*, pp. 375–380, (2014).

[25] Ricardo Gonçalves, Matthias Knorr, and João Leite, 'Minimal change in evolving multi-context systems', in *Procs. of EPIA*, eds., Francisco C. Pereira, Penousal Machado, Ernesto Costa, and Amílcar Cardoso, volume 9273 of *LNCS*, pp. 611–623. Springer, (2015).

[26] Tobias Kaminski, Matthias Knorr, and João Leite, 'Efficient paraconsistent reasoning with ontologies and rules', in *Procs. of IJCAI*, eds., Qiang Yang and Michael Wooldridge, pp. 3098–3105. AAAI Press, (2015).

[27] Matthias Knorr, José Júlio Alferes, and Pascal Hitzler, 'Local closed world reasoning with description logics under the well-founded semantics', *Artif. Intell.*, **175**(9-10), 1528–1554, (2011).

[28] Matthias Knorr, Ricardo Gonçalves, and João Leite, 'On efficient evolving multi-context systems', in *Procs. of PRICAI*, eds., Duc Nghia Pham and Seong-Bae Park, volume 8862 of *LNCS*, pp. 284–296. Springer, (2014).

[29] Freddy Lécué and Jeff Z. Pan, 'Predicting knowledge in an ontology stream', in *Procs. of IJCAI*, ed., Francesca Rossi, pp. 2662–2669. IJCAI/AAAI, (2013).

[30] John McCarthy, 'Generality in artificial intelligence', *Commun. ACM*, **30**(12), 1029–1035, (1987).

[31] Boris Motik and Riccardo Rosati, 'Reconciling description logics and rules', *Journal of the ACM*, **57**(5), 93–154, (2010).

[32] Özgür L. Özçep, Ralf Möller, Christian Neuenstadt, Dmitriy Zheleznayakow, Evgeny Kharlamov, Ian Horrocks, Thomas Hubauer, and Mikhail Roshchin, 'D5.1 Executive Summary: A semantics for temporal and stream-based query answering in an OBDA context', Technical report, Optique FP7-ICT-2011-8-318338 Project, (2013).

[33] Floris Roelofsen and Luciano Serafini, 'Minimal and absent information in contexts', in *Procs. of IJCAI*, eds., Leslie Pack Kaelbling and Alessandro Saffiotti, pp. 558–563. Professional Book Center, (2005).