Multi-Context Systems in Dynamic Environments

Pedro Cabalar · **Stefania Costantini** ·
Giovanni De Gasperis · **Andrea Formisano**

Abstract Multi-Context Systems (MCSs) are able to formally model, in Computational Logic, distributed systems composed of heterogeneous sources, or "contexts", interacting via special rules called "bridge rules". In this paper, we consider how to enhance flexibility and generality in bridge-rules definition and use. In particular, we introduce and discuss some formal extensions of MCSs aimed to their practical application in dynamic environments, and we provide guidelines for implementations.

Keywords Automated Reasoning · Multi-Context Systems · Heterogeneous Distributed Systems

1 Introduction

Multi-Context Systems (MCSs) have been proposed in Artificial Intelligence and Knowledge Representation to model information exchange among heterogeneous

P. Cabalar
University of Corunna, Spain
E-mail: cabalar@udc.es

S. Costantini
Università di L'Aquila, Italy
E-mail: stefania.costantini@univaq.it

G. De Gasperis
Università di L'Aquila, Italy
E-mail: giovanni.degasperis@univaq.it

A. Formisano
Università di Perugia, GNCS-INdAM, Italy
E-mail: andrea.formisano@unipg.it

sources [8,10,12]. MCSs do not make any assumption about such sources nor require them to be homogeneous; rather, the MCS approach deals explicitly with their different representation languages and semantics. These sources, also called *contexts* or *modules*, interact through special interconnection expressions called *bridge rules*; such rules are very similar in syntax and in meaning to logic programming rules as seen in Datalog (cf. [39,3]) or Answer Set Programming (ASP, cf. [32,33,40,31] for foundations. See also [29] and the references therein for applications of ASP), save that atoms in their bodies refer to knowledge items to be obtained from external contexts. A concept of *equilibria* is defined for MCSs, where an equilibrium is a system's state which is "stable" w.r.t. knowledge exchange; interesting system's properties can be defined and safely checked referring to such states.

The reason why MCSs are particularly interesting is that they aim at a formal modeling of real applications requiring access to distributed sources, possibly on the web. In many application domains the adoption of MCSs can bring real advances, whenever different types of heterogeneous information systems are involved and a rigorous formalization should be adopted, also in view of reliability and verifiability of the resulting systems. Notice that this kind of systems often involves agents; MCSs encompassing logical agents have in fact been proposed in the literature (cf., [16]).

Given the potential impact of MCSs for practical knowledge representation and reasoning, there are some aspects in which their definition is still too abstract. In this paper, we introduce and discuss some formal extensions of MCSs useful for a practical use in dynamic environments, and we try to provide guidelines for implementations. The paper considers in particular the following aspects, that will be motivated and discussed in detail after the formal introduction of MCSs:

(i) The specification of how to specialize general bridge rules to the specific application data at hand, and how to trigger their activation upon need; technically, we will cope with bridge-rule grounding and *grounded equilibria*, and we will propose a concept of *proactive* bridge-rule activation.

(ii) The explicit definition of the *evolution* of an MCS over time in consequence of changes to the modules' knowledge bases; in particular, we consider kinds of evolution more general than those already seen in the literature.

(iii) The specification of how to specialize general bridge rules in an evolving MCS, which is a necessary extension of what treated in (i).

(iv) The introduction of "Bridge Rules Patterns", by which we make bridge rules parametric w.r.t. the actual contexts to be queried at run-time; this is done by introducing special terms called *context designators* to be replaced by actual context names dynamically.

(v) The specification of Dynamic MCSs, i.e., MCSs whose composition (in terms of participating contexts) can change over time; in fact, in practical settings context may join or leave the system, or can be unavailable at certain time intervals (e.g., due to system/network faults).

(vi) The introduction of a "Multi-Source option", i.e., whenever a required information can be obtained from several other contexts, a choice may be possible concerning the *preferred* source, where preferences may change over time and according to current circumstances.

(vii) The definition of practical modalities of execution of bridge rules in a distributed MCS.

All these extensions represent substantial and much-needed improvements to the basic MCS framework, as certified by existing connections with related work on MCSs, discussed in next sections, that treat (though in a different way) some of these issues.

This paper is a revised extended version of the work presented in [23, 14], and is organized in the following way. We first formally recall the MCS approach (Section 2). In Section 3, after illustrating significant motivating scenarios, we better discuss points (i)-(vii) introduced above, explaining why the pre-existing definitions are in our opinion problematic in practical application settings, and in which directions they might be improved. In the subsequent sections, for every such aspect we discuss and formalize some variations, enhancements, and extensions to basic MCSs that we deem to be necessary/useful to improve the basic paradigm. In Section 9 we present a case study to illustrate the practical usefulness of the enhanced features. In Section 10 we discuss the complexity of the new framework, and finally we conclude. We have chosen to discuss related work whenever a discussion is useful and pertinent rather than in a separate section.

2 Bridge Rules and Multi-Context Systems

Heterogeneous MCSs have been introduced in [34] in order to integrate different inference systems without resorting to non-classical logic systems. Later, the idea has been further developed and generalized to non-monotonic reasoning domains —see, for instance, [8, 10–12, 42] among many. (Managed) MCSs are designed for the construction of systems that need to access multiple (heterogeneous) data sources called *contexts*. The information flow is modeled via *bridge rules*, whose form is similar to Datalog rules with negation where however each element in their "body" explicitly includes the indication of the contexts from which each item of information is to be obtained. To represent the heterogeneity of sources, each context is supposed to be based on its own *logic*, defined in a very general way [10]. A logic L defines its own syntax as a set F of possible *formulas* (or *KB*-elements) under some signature possibly containing *predicates*, *constants*, and *functions*. As usual, formulas are expressions built upon the idea of *atom*, that is, the application of a predicate to a number n (the predicate arity) of *terms*. The latter, in their turn, can be variables, constants, or compound terms using function symbols, as usual [39]. A *term/atom/formula* is *ground* if there are no variables occurring therein. A logic is *relational* if in its signature the set of function symbols is empty, so its terms are variables and constants only. Formulas can be grouped to form some *knowledge base*, $kb \in 2^F$. The set of all knowledge bases for L is defined as some $KB \subseteq 2^F$. The logic also defines *beliefs* or *data* (usually, ground facts) that can be derived as consequences from a given $kb \in KB$. The set Cn represents all possible belief sets in logic L. Finally, the logic specification must also define some kind of inference or entailment. This is done by defining which belief sets are *acceptable* consequences of a given $kb \in KB$ with a relation $ACC \subseteq KB \times Cn$. Thus, $ACC(kb, S)$ means that belief set S is an acceptable con-

sequence of knowledge base kb. We can also use ACC as a function $ACC : KB \to 2^{Cn}$ where $S \in ACC(kb)$ is the same as $ACC(kb, S)$ as a relation. To sum up, logic L can be completely specified by the triple $\langle KB, Cn, ACC \rangle$.

A *multi-context system* (MCS) $M = \{C_1, \ldots, C_\ell\}$ is a set of $\ell = |M|$ *contexts*, each of them of the form $C_i = \langle c_i, L_i, kb_i, br_i \rangle$, where c_i is the context *name* (unique for each context; if a specific name is omitted, i can act as such), $L_i = \langle KB_i, Cn_i, ACC_i \rangle$ is a logic, $kb_i \in KB_i$ is a knowledge base, and br_i is a set of *bridge rules*. Each bridge rule $\rho \in br_i$ has the form

$$s \leftarrow A_1, \ldots, A_h, not\, A_{h+1}, \ldots, not\, A_m \qquad (1)$$

where the left-hand side s is called the *head*, denoted as $hd(\rho)$, the right-hand side is called the *body*, denoted as $body(\rho)$, and the comma stands for conjunction. We define the *positive* (resp. *negative*) *body* as $body^+(\rho) = \{A_1, \ldots, A_h\}$ (resp. $body^-(\rho) = \{A_{h+1}, \ldots, A_m\}$). The head $hd(\rho) = s$ is a formula in L_i such that $(kb_i \cup \{s\}) \in KB_i$. Each element A_k in the body has the form $(c_j : p)$ for a given j, $1 \le j \le |M|$, and can be seen as a *query* to the context $C_j \in M$ (possibly different from C_i) whose name is c_j, to check the status of belief p (a formula from L_j) with respect to the current belief state (defined below) in C_j. When the query is made in the context $j = i$ we will omit the context name, simply writing p instead of $(c_i : p)$. In case the entire body "succeeds" then the bridge rule can be applied, and its conclusion can be exploited within context c_i.

A *belief state* (or *data state*) $\mathbf{S}$ of an MCS M is a tuple $\mathbf{S} = (S_1, \ldots, S_\ell)$ such that for $1 \le i \le \ell = |M|$, $S_i \in Cn_i$. Thus, a data state associates to each context C_i a possible set of consequences S_i. A bridge rule $\rho \in br_i$ of context $C_i \in M$ is *applicable* in belief state $\mathbf{S}$ when, for any $(c_j : p_j) \in body^+(\rho)$, $p_j \in S_j$, and for any $(c_k : p_k) \in body^-(\rho)$, $p_k \notin S_k$; so, $app(\mathbf{S})$ is the set the heads of those bridge rules that are *applicable* in $\mathbf{S}$.

In managed MCSs (mMCSs)[1] the conclusion s, which represents the "bare" bridge-rule result, becomes $o(s)$, where o is a special operator. The meaning is that s is processed by operator o, that can perform any elaboration, such as format conversion, belief revision, etc. More precisely, for a given logic L with formulas $F = \{s \in kb \mid kb \in KB\}$, a *management base* is a set of operation names (briefly, operations) OP, defining elaborations that can be performed on formulas. For a logic L and a management base OP, the set of operational statements that can be built from OP and F is $F^{OP} = \{o(s) \mid o \in OP, s \in F\}$. The semantics of such statements is given by a *management function*, $mng : 2^{F^{OP}} \times KB \to KB$, which maps a set of operational statements and a knowledge base into a (possibly different) modified knowledge base.[2] Now, each context $C_i = \langle c_i, L_i, kb_i, br_i, OP_i, mng_i \rangle$ in an mMCS is extended to include its own management function mng_i which is crucial for knowledge incorporation from external sources. Notice that, management functions are not required to be monotonic operators.

The application of the management function mng_i to the result of the applicable rules must be acceptable with respect to ACC_i. We say that a belief state $\mathbf{S} = $

---

[1] For the sake of simplicity, we define mMCS simply over logics instead of "logic suite" as done in [11], where one can select the desired semantics among a set of possibilities.

[2] We assume a management function to be deterministic, i.e., to produce a unique new knowledge base.

$(S_1, \ldots, S_\ell)$ is an *equilibrium* for an mMCS M iff, for $1 \leq i \leq \ell$,

$$S_i \in ACC_i(mng_i(app(\mathbf{S}), kb_i)) \tag{2}$$

Namely, for each context C_i, one

  (i) applies all the bridge rules of C_i that are applicable in the belief state $\mathbf{S}$ (namely, obtains the set $app(\mathbf{S})$);
 (ii) applies the management function which, by incorporating bridge-rule results into C_i's knowledge base kb_i, computes a new knowledge base $kb'_i = mng_i(app(\mathbf{S}), kb_i)$;
(iii) determines via ACC_i the set of acceptable sets of consequences of kb'_i.

In an equilibrium, such a set includes S_i, i.e., an equilibrium is "stable" w.r.t. bridge-rule application.

Conditions for existence of equilibria and the complexity of deciding equilibrium existence for mMCS have been studied [8]; roughly speaking, such complexity depends on the complexity of computing formula (2) for each $C_i \in M$. Algorithms for computing equilibria have been recently proposed [10,26] but, in practice, they are only applicable when open-access to contexts' contents is granted. For practical purposes, one often provisionally assume that equilibria exist, and that they do not include inconsistent data sets. It has been proved that such *local consistency* is achieved whenever all management functions are (lc-)preserving, i.e., if they always determine a kb' which is consistent.

3 Motivating Scenarios and Discussion

Some of the reasons of our interest in (m)MCSs and bridge-rules stem from a project where we are among the proponents [1], concerning smart Cyber Physical Systems, with particular attention (though without restriction) to applications in the e-Health field. The general scenario of such "F&K" ("Friendly-and-Kind") systems [1] is depicted in Figure 1. In such setting we have a set of computational entities, of knowledge bases, and of sensors, all immersed in the "Fog" of the Internet of Things. All components can, in time, join or leave the system. Some computational components will be agents. In the envisaged e-Health application for instance, an agent will be in charge of each patient. Such an agent will interact with other computational entities, with the patient, and with the environment, possibly exploiting sensors and other communication capabilities (e.g., image and speech recognition, etc). The System's engine will keep track of present system's configuration, and will enable the various classes of rules to work properly. Terminological rules will allow for more flexible knowledge exchange via Ontologies. Pattern Rules will have the role of defining and checking coherence/correctness of system's behavior. Bridge rules are the vital element, as they allow knowledge to flow among components in a clearly-specified principled way: referring to Figure 1, devices for bridge-rule functioning can be considered as a part of the System's engine. Therefore, F&Ks are "knowledge-intensive" systems, providing flexible access to dynamic, heterogeneous, and distributed sources of knowledge and reasoning, within a highly dynamic computational environment.

ENVISAGED SMART CPS GENERAL ARCHITECTURE
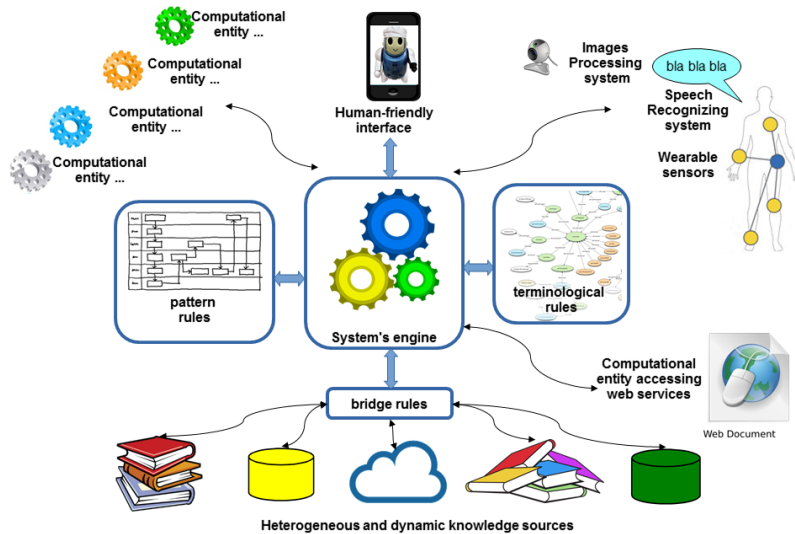
Fig. 1 Motivating Scenario: general architecture of the "Friendly-and-Kind" Cyber Physical System [1]. An agent takes case of a patient by monitoring her health status and by proactively interacting with other computational entities and knowledge sources in case of need.

We basically consider such systems to be (enhanced) mMCSs: as mentioned in fact, suitable extensions to include agents and sensors in such systems already exist.

Another application (depicted in Figure 2) in a very different domain though with some analogous features is aimed at Digital Forensics and Digital Investigations. Digital Forensics is a branch of criminalistics which deals with the identification, acquisition, preservation, analysis, and presentation of the information content of computer systems, or in general of digital devices, in relation to crimes that have been perpetrated. The objective of the Evidence Analysis stage of Digital Forensics, or more generally of Digital Investigations, is to identify, categorize, and formalize digital sources of evidence (or, however, sources of evidence represented in digital form). The objective is to organize such sources of proof into evidences, so as to make them robust in view of their discussion in court, either in civil or penal trials.

In recent work[3] [24,41] we have identified a setting where an intelligent agent is in charge of supporting the human investigator in such activities. The agent should help identify, retrieve, and gather the various kinds of potentially useful evidence, process them via suitable reasoning modules, and integrate the results into coherent evidence. In this task, the agent may need to retrieve and exploit knowledge bases concerning, e.g., legislation, past cases, suspect's criminal history, and so on. In the

[3] Supported by Action COST CA17124 "DIGFORASP: DIGital FORensics: evidence Analysis via intelligent Systems and Practices, start September 10, 2018."
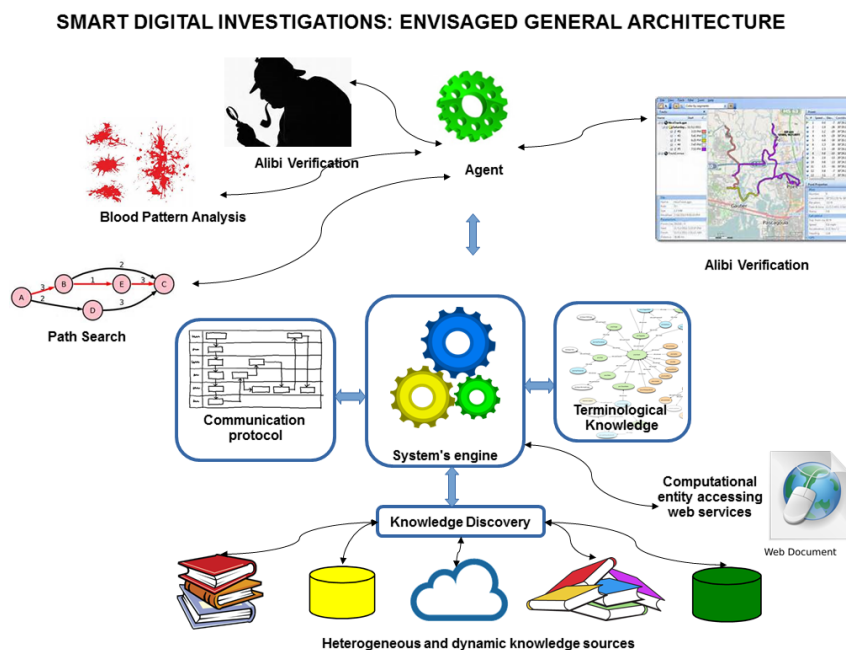
Fig. 2 General architecture of a MCS aimed at supporting Digital Forensics and Digital Investigations. The system assists human activities by enabling access, interaction, and cross integration between computational entities, sources of knowledge, and sources of evidences.

picture, the agent considers: results from blood-pattern analysis on the crime scene, which lead to model such a scene via a graph, where suitable graph reasoning may reconstruct the possible patterns of action of the murderer; alibi verification in the sense of a check of the GPS positions of suspects, so as to ascertain the possibility of her/him being present on the crime scene on the crime time; alibi verification in the sense of double-checking the suspect's declarations with digital data such as computer logs, videos from video cameras situated on the suspect's path, etc. All the above can be integrated with further evidence such as the results of DNA analysis and others. The system can also include Complex Event Processing so as to infer from significant clues the possibility that a crime is being or will be perpetrated.

In our view also this system can be seen as an (enhanced) mMCSs. In reality however, many of the involved data must be retrieved, elaborated, or checked from knowledge bases belonging to organizations which are external to the agent, and have their own rules and regulations for data access and data elaboration. Thus, suitable modalities for data retrieval and integration must be established in the agent to correctly access such organizations. Therefore, a relevant issue that we mention but we do not treat here is exactly that of modalities of access to contexts included in an MCS, which possibly include time limitations and the payment of fees.

In the perspective of the definition and implementation of such kind of systems as mMCSs, the definition recalled in Section 2 must be somehow enhanced. Our

first experiments show in fact that such definition is, though neat, too abstract when confronted with practical implementation issues.

We list below and discuss some aspects that constitute limitations to the actual flexible applicability of the MCS paradigm, or that might anyway be usefully improved.

For the sake of illustration, we consider throughout the rest of the paper two running examples. The first example has been introduced in [21], where we have proposed Agent Computational Environments (ACE), that are actually MCSs including agents among their components. In [21], an ACE has been used to model a personal assistant agent aiding a prospective college student in choosing and applying to an university, taking into account the subjects of interest, the available budget, and other preferences. The second example concerns F&Ks, and is developed in detail in Section 9 where we propose a full though small case study. There, a patient can be in need of a medical doctor, possibly a specialized one, and may not know in advance which one to consult. In the examples we adopt a Prolog-like syntax [39, 3], with constant, predicate and function symbols indicated by names starting with a lowercase letter, and variable symbols indicated by names starting with an uppercase letter.

Grounded Knowledge Assumption. Bridge rules are ground by definition, i.e., they do not contain variables. In [11] it is literally stated that [in their examples] they "*use for readability and succinctness schematic bridge rules with variables (upper case letters and '_' [the 'anonymous' variable]) which range over associated sets of constants; they stand for all respective instances (obtainable by value substitution)*". The basic definition of mMCS does not require either contexts' knowledge bases or bridge rules to be finite sets. Though contexts' knowledge bases will in practice be finite, they cannot be assumed to necessarily admit a finite grounding, and thus a finite number of bridge-rules' ground instances. This assumption can be reasonable, e.g., for standard relational databases and logic programming under the answer set semantics [31]. In other kinds of logics, for instance simply "plain" general logic programs, it is no longer realistic. In practical applications however, there should either be a finite number of applicable (ground instances of) bridge-rules, or some suitable device for run-time dynamic bridge-rule instantiation and application should be provided. To emphasize the usefulness of non-ground bridge rules, let us consider our running examples.

Example 1 Suppose that a patient is looking for a cardiologist from a medical directory, represented as a context, say, e.g., called *med-dir*. A ground bridge rule might look like:

$find\_cardiologist(maggie\text{-}smith) \leftarrow$
     $med\text{-}dir : cardiologist(maggie\text{-}smith)$

where the patient would confirm what she already knows, i.e., the rule verifies that Dr. Maggie Smith is actually listed in the directory.
Instead, a patient may in general intend to use a rule such as:

$find\_cardiologist(N) \leftarrow$
     $med\text{-}dir : cardiologist(N)$

where the query to the medical directory will return in variable *N* the name (e.g., Maggie Smith) of a previously unknown cardiologist. Let us assume that *N* is instantiated to *maggie-smith* which is the name of a context representing the cardiologist's personal assistant.

Similarly, a prospective student's personal assistant will query universities about the courses that they propose, where this information is new and will be obtained via a bridge rule which is not ground.   □

Notice that for actual enrichment of a context's knowledge one must allow *value invention*, that is, a constant returned via application of a non-ground bridge rule may not previously occur in the destination module; in this way, a module can "learn" really new information through inter-context exchange.

Update Problem. Considering inputs from sensor networks as done in [12] is a starting point to make MCSs evolve in time and to have contexts which update their knowledge base and thus cope flexibly with a changing environment. However, sources can be updated in many ways via the interaction with their environment. For instance, agents are supposed to continuously modify themselves via sensing and communication with other agents, but even a plain relational database can be modified by its users/administrators. Referring to the examples, a medical directory will be updated periodically, and universities will occasionally change the set of courses that they offer, update the fees and other features of interest to students. Where one might adopt fictitious sensors (as suggested in relevant literature) in order to simulate many kinds of updates, a more general update mechanism seems in order. Such mechanism should assume that each context has its own update operators and its own modalities of application. An MCS can be "parametric" w.r.t. contexts' update operators as it is parametric w.r.t. contexts' logics and management functions.

Logical Omniscience Assumption and Bridge Rules Application Mechanisms. In MCS, bridge rules are supposed to be applied whenever their body is entailed by the current data state. In fact bridge rules are, in the original MCS formulation, a reactive device where each bridge rule is applied whenever applicable. In reality, contexts are in general not logical omniscient and will hardly compute their full set of consequences beforehand. So, the set of bridge rules which are applicable at each stage is not fully known. Thus, practical bridge rule application will presumably consist in an *attempt of application*, performed by posing queries, corresponding to the elements of the body of the bridge rule, to other contexts which are situated somewhere in the nodes of a distributed system. The queried contexts will often determine the required answer upon request. Each source will need time to compute and deliver the required result and might even never be able to do so, in case of reasoning with limited resources or of network failures. Moreover, contexts may want to apply bridge rules only if and when their results are needed. So, a generalization of bridge-rule applicability in order to make it *proactive* rather than reactive can indeed be useful. In our working example, for instance, a patient might look for a cardiologist (by enabling the bridge rule seen above) only if some health condition makes it necessary.

Static Set of Bridge Rules. Equipping a context with a static set of bridge rules can be a limitation; in fact, in such bridge rules all contexts to be queried are fully known in advance. In contexts' practical reasoning instead, it can become known only at "runtime" which are the specific contexts to be queried in the situations that practically arise. To enhance flexibility in this sense, we introduce *Bridge Rules Patterns* to make bridge rules parametric w.r.t. the contexts to be queried; such patterns are meta-level rule schemata where in place of contexts' names we introduce special terms called *context designators*. Bridge rule patterns can be specialized to actual bridge rules by choosing which specific contexts (of a certain kind) should specifically be queried in the situation at hand. This is a very important extension which avoid a designer's omniscience about how the system will evolve.

Example 2 After acquiring as seen above (Examples 1), the reference to a reliable cardiologist (i.e., *maggie-smith*), the patient (say Mary) can get in contact with the cardiologist, disclose her health condition C, and thus make an appointment for time T. This, in our approach, can be made by taking a general bridge rule

$make\_appointment(mary, T) \leftarrow$
        $condition(mary, C),$
        $mycardiologist(c) : consultation\_needed(mary, C, T)$

where $mycardiologist(c)$ is a *context designator*, that is a placeholder intended to denote a context not yet known. Such *bridge rule pattern* can be instantiated, at runtime, to the specific bridge rule

$make\_appointment(mary, T) \leftarrow$
        $condition(mary, C),$
        $maggie\text{-}smith : consultation\_needed(mary, C, T)$

$\square$

Static System Assumption. The definition of mMCS might realistically be extended in order to encompass settings where the set of contexts changes over time. This to take into account dynamic aspects such as momentarily disconnections of contexts or the fact that components may freely either join or abandon the system or that inter-context reachability might change over time.

Full System Knowledge Assumption and Unique Source Assumption. A context might know the *role* of another context it wants to query (e.g., a medical directory, or diagnostic knowledge base) but not its "name" that could be, for instance, its URI or anyway some kind of reference that allows for actually posing a query. In the body of bridge rules, each literal mentions however a specific context (even for bridge rule patterns, context designators must be instantiated to specific context names). We enrich the MCS definition with a directory facility, where instead of context names bridge rules may employ queries to the directory to obtain contexts with the required role. Each "destination" context might have its own *preferences* among contexts with the same role. We also introduce a structure in the system, by allowing to specify reachability, i.e., which context can query which one. This corresponds to the nature of many applications: a context can sometimes be not allowed to access all or some of the others, either for a matter of security/privacy or for a matter of convenience.

Often, information must be obtained via suitable mediation, while access to every information source is not necessarily either allowed or desirable.

Uniform Knowledge Representation Format Assumption. Different contexts might represent similar concepts in different ways: this aspect is taken into account in [17] (and so it is not treated here), where ontological definitions can be exchanged among contexts and a possible global ontology is also considered.

Equilibria Computation and Consistency Check Assumption. As mentioned earlier, algorithms for computing equilibria are practically applicable only if open access to contexts' contents is granted. The same holds for local and global consistency checking. However, the potential of MCSs is, in our view, that of modeling real distributed systems where contexts in general keep their knowledge bases private. Therefore, in practice, one will often just assume the existence of consistent equilibria. This problem is not treated here, but deserves due attention for devising interesting sufficient conditions.

In the next sections we discuss each aspect and we introduce our proposals of improvement. We devised the proposed extensions in the perspective of real applications of mMCSs. In view of such applications, we realized that issues related to the concrete modalities of bridge-rule instantiation, activation, and execution are the first ones that needed to be considered. We will, in fact, present the proposed enhancements to the MCSs formalization in an incremental fashion; whenever we introduce new definitions, they are formulated so as to be applicable in the setting specified by the previous ones.

4 Grounded Knowledge Assumption

To the best of our knowledge, the problem of loosening the constraint of bridge-rules groundedness has not been so far extensively treated in the literature and no satisfactory solution exists. The issue of bridge-rule grounding has been discussed in [4] for *relational* MCSs [30], which are a variant of MCSs where all the involved logics are relational. In [4], however, grounding is performed over a carefully defined finite domain, composed of constants only. Instead, we intend to consider any, even infinite, domain.

In the rest of this section we first provide by examples an intuitive idea of how grounding might be performed, step by step, along with the computation of equilibria. Then, we provide a formalization of our ideas.

4.1 Intuition

The procedure for computing equilibria that we propose for the case of non-ground bridge rules is, informally, the following.

(i) We consider a standard initial data state $\mathbf{S}_0$.

(ii) We instantiate bridge rules over ground terms occurring in S_0; we thus obtain an initial bridge-rule grounding relative to S_0.

(iii) We evaluate whether S_0 is an equilibrium, i.e., if S_0 coincides with the data state S_1 resulting from applicable bridge rules.

(iv) In case S_0 is not an equilibrium, bridge rules can now be grounded w.r.t. terms occurring in S_1, and so on, until either an equilibrium is reached, or no more applicable bridge rules are generated.

It is reasonable to establish the initial data state S_0, from where to start the procedure, on the basis of a "raw" data state where each element consists of the set of ground atoms obtained from the initial knowledge base of each context. That is, we consider each context's Herbrand Universe [39]: one takes functions, predicate symbols and constants occurring in the context's knowledge base and builds data state items by constructing all possible ground atoms. By definition, a ground instance of a context C_i's knowledge base is in fact an element of Cn_i, i.e., it is a set of possible consequences, though in general it is not acceptable. An acceptable S_0 can be obtained from such "raw" data state by applying to each of its elements the relative context's acceptability function, that here we assume to return a single set.

Notice that starting the procedure from S_0 (even when each element of S_0 is a finite set) does not guarantee neither the existence of a finite equilibrium, nor that an equilibrium can be reached in a finite number of steps.

Before introducing two examples to clarify this point, let us require a minor modification in bridge-rule syntax that we will adopt in the rest of the paper; namely, we assume the customary Prolog syntax and procedural semantics, where elements in the body of a rule are proved/executed in left-to-right order.

Example 3 Consider an MCS composed of two contexts C_1 and C_2, both concerning the representation of natural numbers. Assume such contexts to be characterized respectively by the following knowledge bases and bridge rules (where C_1 has no bridge rule).

%kb₁
    $nat(0).$
%kb₂
    $nat(succ(X)) \leftarrow nat(X).$
%br₂
    $nat(X) \leftarrow (c1 : nat(X)).$

The basic data state is $S_0 = (\{nat(0)\}, \emptyset)$, where in fact C_2's initial data state is empty because there is no constant occurring in kb_2. The unique equilibrium is reached in one step via the application of br_2 which "communicates" fact $nat(0)$ to C_2. In fact, due to the recursive rule, we have the equilibrium (S_1, S_2) where $S_1 = \{nat(0)\}$ and $S_2 = \{nat(0), nat(succ(0)), nat(succ(succ(0))), \ldots\}\rangle$ I.e., S_2 is an infinite set representing all natural numbers. If we assume to add a third context C_3 with empty knowledge base and a bridge rule br_3 defined as $nat(X) \leftarrow (c2 : nat(X))$, then the equilibrium would be (S_1, S_2, S_3) with $S_3 = S_2$. There, in fact, br_3 would be grounded on the infinite domain of the terms occurring in S_2, thus admitting an infinite number of instances. $\qquad\square$

Example 4 Consider a variation of the MCS in Example 3 where a context C_1 "produces" the even natural numbers (starting from 0) and a context C_2 produces the odd ones. In this case kb_2 is empty:

$\%kb_1$

    $nat(0)$.

$\%br_1$

    $nat(succ(X)) \leftarrow (c2 : nat(X))$.

$\%br_2$

    $nat(succ(X)) \leftarrow (c1 : nat(X))$.

We may notice that the contexts in the above example enlarge their knowledge by means of mutual "cooperation". Let us consider again, according to our proposed method, the basic data state $\mathbf{S}_0 = (\{nat(0)\}, \emptyset)$. As stated above, bridge rules are grounded on the terms occurring therein. $\mathbf{S}_0$ is not an equilibrium for the given MCS. In fact, the bridge rule in br_2, once grounded on the constant 0, is applicable but not applied. The data set resulting from the application, namely, $\mathbf{S}' = (\{nat(0)\}, \{nat(succ(0))\})$ is not an equilibrium either, because now the bridge rule in br_1 (grounded on $succ(0)$) is in turn applicable but not applied. We may go on, as $\mathbf{S}'' = (\{nat(0), nat(succ(succ(0)))\}, \{nat(succ(0))\})$ leaves the bridge rule in br_2 to be applied (grounded on $succ(succ(0))$), and so on. There is clearly a unique equilibrium that cannot however be reached within finite time, though at each step we have a data state composed of finite sets. The unique equilibrium (reached after a denumerably infinite number of steps), is composed of two infinite sets, the former one representing the even natural numbers (including zero) and the latter representing the odd natural numbers. The equilibrium may be represented as:

$$\mathbf{E} = \big(\{nat(0), nat(succ^k(0)), \ k \bmod 2 \ = 0\}, \ \{nat(succ^k(0)), \ k \bmod 2 \ = 1\}\big)$$

□

We have actually devised and applied an adaptation to non-ground bridge rules of the operational characterization introduced in [8] for the grounded equilibrium of a *definite* MCS. In fact, according to the conditions stated therein C_1 and C_2 are monotonic and admit at each step a unique set of consequences and bridge-rule application is not unfounded (cyclic).

In our more general setting however, the set of ground bridge rules associated to given knowledge bases cannot be computed beforehand and the step-by-step computation must take contexts interactions into account.

Since reaching equilibria finitely may have advantages in practical cases, we show below a suitable reformulation of the above example, that sets via a practical expedient a bound on the number of steps. The equilibrium reached will be partial, in the sense of representing a subset of natural numbers, but can be reached finitely and is composed of finite sets.

Example 5 Consider the following reformulation of the MCS described in Example 4. The knowledge bases and bridge rules now are:[4]

---

[4] Recall that, we are assuming that whenever in some element the body of a bridge rule the context is omitted, i.e., we have just p_j instead of $(c_j : p_j)$, then we require that p_j is proved locally from the present context's knowledge base.

$\%kb_1$
  $nat(0).$
  $count(0).$
  $threshold(t).$
$\%br_1$
  $new(nat(succ(X))) \mathbin{:\!-} count(C), threshold(T), C < T, (c2 : nat(X)).$
$\%kb_2$
  $count(0).$
  $threshold(t).$
$\%br_2$
  $new(nat(succ(X))) \mathbin{:\!-} count(C), threshold(T), C < T, (c1 : nat(X)).$

In the new definition there is a counter (initialized to zero) and some threshold, say t. We exploit a management function that suitably defines the operator *new* which is now applied to bridge-rule results. A logic programming definition of such management function might be the following, where the counter is incremented and the new natural number asserted. Notice that such definition is by no means not logical, as we can shift to the "evolving logic programming" extension [2].

  $new(nat(Z)) \mathbin{:\!-} assert(nat(Z)), increment(C).$
  $increment(C) \mathbin{:\!-} retract(count(C)), C1 \text{ is } C + 1, assert(count(C1)).$

Consequently, bridge rules will now produce a result only until the counter reaches the threshold, which guarantees the existence of a finite equilibrium.   □

4.2 Formalization

Below we formalize the procedure that we have empirically illustrated via the examples, so as to generalize to mMCS with non-ground bridge rules the operational characterization of [8] for monotonic MCSs (i.e., those where each context's knowledge base admits a single set of consequences, which grows monotonically when information is added to the context's knowledge base). Following [8], for simplicity we assume each bridge-rule body to include only positive literals, and the formula s in its head $o(s)$ to be an atom. So, we will be able to introduce the definition of *grounded equilibrium of grade* κ. Preliminarily, in order to admit non-ground bridge rules we have to specify how we obtain their ground instances and how to establish applicability.

Definition 1 Let $r \in br_i$ be a non-ground bridge rule occurring in context C_i of a given mMCS M with belief state $\mathbf{S}$. A ground instance ρ of r w.r.t. $\mathbf{S}$ is obtained by substituting every variable occurring in r (i.e., occurring either in the elements $(c_j : p_j)$ in the body of r or in its head $o(s)$ or in both) via (ground) terms occurring in $\mathbf{S}$.

For an mMCS M, a data state $\mathbf{S}$ and a ground bridge rule ρ, let $app^{\models_g}(\rho, \mathbf{S})$ be a Boolean-valued function which checks, in the ground case, bridge-rule body entailment w.r.t. $\mathbf{S}$. Let us redefine bridge-rule applicability.

Definition 2 The set $app(S)$ relative to ground bridge rules which are applicable in a data state $\mathbf{S}$ of a given mMCS $M = (C_1, \ldots, C_\ell)$ is now defined as follows.

$app(\mathbf{S}) = \big\{hd(\rho) \mid \rho$ is a ground instance w.r.t. $\mathbf{S}$ of some

   bridge rule $r \in br_i, 1 \leq i \leq \ell,$ and $app^{\models_g}(\rho, \mathbf{S}) = true\big\}$

We assume, analogously to [8], that a given mMCS is *monotonic*, which here means that for each C_i the following properties hold:

(i) ACC_i is monotonic w.r.t. additions to the context's knowledge base, and
(ii) mng_i is monotonic, i.e., it allows to only add formulas to C_i's knowledge base.

Let, for a context C_i, the function ACC_i' be a variation of ACC_i which selects one single set E_i among those generated by ACC_i and let ACC_i' be monotonic. Namely, given a context C_i and a knowledge base $\hat{kb} \in KB_{L_i}$, $ACC_i'(\hat{kb}) = E_i$ where $E_i \in ACC_i(\hat{kb})$. Let, moreover, ∞ be the first infinite ordinal number isomorphic to the natural numbers. We introduce the following definition:

Definition 3 Let $M = (C_1, \ldots, C_\ell)$ be an mMCS with no negative literals in bridge-rule bodies, and assume arbitrary choice of function ACC_i' for each composing context C_i. Let, for each $1 \leq i \leq \ell$, $gr(kb_i)$ be the grounding of kb_i w.r.t. the constants occurring in any kb_j, for $1 \leq j \leq \ell$. A *data state of grade* κ is obtained as follows.

- For $i \leq \ell$, we let $kb_i^0 = gr(kb_i)$.

- For each $\alpha \geq 0$, we let $\mathbf{S}^\alpha = (S_1^\alpha, \ldots, S_\ell^\alpha)$ with $S_i^\alpha = ACC_i'(kb_i^\alpha)$ and where, for finite κ we have
$$kb_i^{\alpha+1} = \begin{cases} mng_i(app(\mathbf{S}^\alpha), kb_i^\alpha) \text{ if } \alpha < \kappa, \\ kb_i^\alpha \text{ otherwise} \end{cases}$$
while, if $\kappa = \infty$ we put $kb_i^\infty = \bigcup_{\alpha \geq 0} kb_i^\alpha$.

Example 6 The grade κ of a data state computed by the above formula takes the role of the threshold used in Example 5. Referring, in fact, to the MCS of Example 4, by setting $\kappa = 1$ we would compute $\mathbf{S}'$, while for $\kappa = 2$ we would obtain $\mathbf{S}''$.   □

Differently from [8], the computation of a new data state element is provided here according to mMCSs, and thus involves the application of the management function to the present knowledge base so as to obtain a new one. Such data state element is then the unique set of consequences of the new knowledge base, as computed by the ACC_i' function.

The result can be an equilibrium only if the specified grade is sufficient to account for all potential bridge-rules applications. In the terminology of [8] it would then be a *grounded equilibrium*, as it is computed iteratively and deterministically from the contexts' initial knowledge bases. We have the following.

Definition 4 Let $M = (C_1, \ldots, C_\ell)$ be a monotonic mMCS with no negative literals in bridge-rule bodies. A belief state $\mathbf{S} = (S_1, \ldots, S_\ell)$ is a grounded equilibrium of grade κ of M iff $ACC_i'(mng_i(app(\mathbf{S}), kb_i^\kappa)) = S_i$, for $1 \leq i \leq \ell$.

Proposition 1 *Let* $M = (C_1, \ldots, C_\ell)$ *be a monotonic mMCS with no negative literals in bridge-rule bodies. A belief state* $\mathbf{S} = (S_1, \ldots, S_\ell)$ *is a grounded equilibrium of grade* ∞ *(or simply 'a grounded equilibrium') for M iff* $\mathbf{S}$ *is an equilibrium for M.*

Proof **S** is an equilibrium for M because it fulfills the definition by construction. In fact, step-by-step all applicable bridge rules will have been applied, so **S**, obtained via ∞ steps, is stable w.r.t. bridge rule application. The converse is straightforward. $\qquad\square$

Notice that reaching a grounded equilibrium of grade ∞ does not always require an infinite number of steps: the procedure of Definition 3 can reach a fixpoint in a number δ of steps, where either $\delta = \infty$ or δ is finite. In fact, the required grade for obtaining an equilibrium would be $\kappa = \infty$ in the former version of our example, where in the latter version if setting threshold t we would have $\kappa = t$.

Several grounded equilibria may exist, depending upon the choice of ACC_i'. We can state the following relationship with [8]:

Proposition 2 *Let $M = (C_1, \ldots, C_\ell)$ be a definite MCS (in the sense of [8]), and let* **S** $= (S_1, \ldots, S_\ell)$ *be a grounded equilibrium for M according to their definition. Then,* **S** *is a grounded equilibrium of the mMCS M' obtained by including in M the same contexts as in M, and, for each context C_i, letting $ACC_i' = ACC_i$, and associating to C_i a management function mng_i that just adds to kb_i every s such that $o(s) \in app(\mathbf{S})$.*

Proof As all the bridge rules in both M and M' are ground, the procedure of Definition 3 and the procedure described on page 4 (below Definition 11) in [8] become identical, as we added only two aspects, i.e., considering non-ground bridge rules, and considering a management function, which are not applicable to M'. $\qquad\square$

In [8], where the authors consider *ground* bridge rules only, they are able to transfer the concept of grounded equilibrium of grade ∞ of a monotonic MCS M to its extensions, where an extension is defined below. Intuitively, an extension M' of (m)MCS M has the same number of contexts, each context in M' has the same knowledge base of the corresponding context in M, but, possibly, a wider set of bridge rules; some of these bridge rules, however, *extend* those in M as they agree on the positive bodies.

Definition 5 Let $M = (C_1, \ldots, C_\ell)$ be a monotonic (m)MCS with no negative literals in bridge-rule bodies. A monotonic (m)MCS $M' = (C_1', \ldots, C_\ell')$ is an *extension* of M iff for each $C_i = \langle c_i, L_i, kb_i, br_i, OP_i, mng_i \rangle$, we have that $C_i' = \langle c_i, L_i, kb_i, br_i', OP_i, mng_i \rangle$ where $\rho \in br_i$ implies $\exists \rho' \in br_i'$ where $body^+(\rho) = body^+(\rho')$ and $body^-(\rho')$ may be nonempty. We call ρ and ρ' *corresponding* bridge rules.

In [8] it is stated that a grounded equilibrium **S** $= (S_1, \ldots, S_\ell)$ of grade ∞ of a definite MCS M is a grounded equilibrium of grade ∞ of any extension M' of M that can be obtained from M' by: canceling all bridge rules where **S** does not imply the negative body, and canceling the negative body of all remaining bridge rules. We may notice that [8] proceeds from M' to M via a reduction similar to the Gelfond-Lifschitz reduction [32]. In our case, we assume to extend the procedure of Definition 3 by dropping the assumption of the absence of negation in bridge-rule bodies. So, the procedure will now be applicable to monotonic MCSs in general. Then, we have that:

Proposition 3 *Let $M = (C_1, \ldots, C_\ell)$ be a monotonic (m)MCS with no negative literals in bridge-rule bodies, and let M' be an extension of M. Let* **S** $= (S_1, \ldots, S_\ell)$ *be a*

*grounded equilibrium for M, reachable in δ steps. **S** is a grounded equilibrium for M' iff, if applying the procedure of Definition 3 to both M and M', we have that $\forall \alpha > 0$, $\rho \in app(\mathbf{S}^{\alpha})$ iff $\rho' \in app(\mathbf{S'}^{\alpha})$, where ρ and ρ' are corresponding bridge rules **S** and **S'** are the data states at step α of M and M', respectively.*

Proof The result follows immediately from the fact that at each step corresponding bridge rules are applied in *M* and *M'* on the same knowledge bases, so at step ∞ the same belief state will have been computed. A proof by induction is thus straightforward. □

The extension of the above results to the case of general non-monotonic MCSs will be subject of further research.

5 Update Problem: Update Operators and Timed Equilibria

Bridge rules as defined in mMCSs are basically a *reactive* device, as a bridge rule is applied whenever applicable. In dynamic environments, however, a bridge rule in general will not be applied only once, and it does not hold that an equilibrium, once reached, lasts forever. In fact, in recent extensions to mMCS, contexts are able to incorporate new data items coming from observations, so that a bridge rule can be, in principle, *re-evaluated* upon new observations. For this purpose, two main approaches have been proposed for this type of mMCSs: so-called *reactive* [12] and *evolving* [37] multi-context systems. A *reactive* MCS (rMCS) is an mMCS where the system is supposed to be equipped with a set of sensors that can provide observations from a set *Obs*. Bridge rules can refer now to a new type of atoms of the form $o@s$, being $o \in Obs$ an observation that can be provided by sensor *s*. A "run" of such system starts from an equilibrium and consists in a sequence of equilibria induced by a sequence of sets of observations. In an *evolving* MCS (eMCS), instead, there are special *observation contexts* where the observations made over time may cause changes in each context knowledge base. As for the representation of time, different solutions have been proposed. For instance, [12] defines a special context whose belief sets on a run specify a time sequence. Other possibilities assume an "objective" time provided by a particular sensor, or a special head predicate in bridge rules whose unique role is adding timed information to a context.

However, in the general case of dynamic environments, contexts can be realistically supposed to be able to incorporate new data items in several ways, including interaction with a user and with the environment. We thus intend to propose extensions to explicitly take into account not only observations but, more generally, the interaction of contexts with an external environment. Such an interaction needs not to be limited to bridge rules, but can more generally occur as part of the context's reasoning/inference process. We do not claim that the interaction with an external environment *cannot* somehow be expressed via the existing approaches to MCSs which evolve in time [12,37,36]; in fact, as mentioned above, this can be obtained by introducing special kinds of contexts. However, in view of practical applications, we design an *explicit* device to cope with this important issue.

We proceed next to define *timed* MCS (tmMCS). In a tmMCS, the main new feature is the idea of *(update) action*. For each context C_i, we define a set act_i of *elementary actions*, where each element $\pi \in act_i$ is the name of some action or operation that can be performed to update context C_i. We allow a subset $Obs_i \subseteq act_i$ for observing sensor inputs as in [12]. A *compound action* (*action*, for short) Π_i is a set $\Pi_i \subseteq act_i$ of elementary actions that can be simultaneously applied for context C_i. The application of an action to a knowledge base is ruled by an *update function*:

$$\mathscr{U}_i : KB_i \times 2^{act_i} \to 2^{KB_i} \setminus \{\emptyset\},$$

so that $kb_i' \in \mathscr{U}_i(kb_i, \Pi_i)$ means that the "new" knowledge base kb_i' is a possible result of updating kb_i with action Π_i under function $\mathscr{U}_i$. We thus assume $\mathscr{U}_i$ to encompass all possible updates performed to a module (including, for instance, sensor inputs, updates performed by a user, consequences of messages from other agents or changes determined by the context's self re-organization). Note that $\mathscr{U}_i(kb_i, \Pi_i)$ is (possibly) non-deterministic, but never empty (some resulting kb_i' is always specified). In some cases, we may even leave the knowledge base unaltered, since it is admitted that $kb_i \in \mathscr{U}_i(kb_i, \Pi_i)$. Notice, moreover, that update operations can be non-monotonic.

In order to allow arbitrary sequences of updates, we assume a linear time represented by a sequence of *time points* $\{t_T\}_{T \geq 0}$ indexed by natural numbers $T \in \mathbb{N}$ and representing discrete states or instants in which the system is updated. The elapsed time between each pair of states is some real quantity $\delta = t_{T+1} - t_T > 0$. Given $T \in \mathbb{N}$ and context C_i, we write $kb_i[T]$ and $\Pi_i[T]$ to respectively stand for the knowledge base content at instant T and the action performed to update C_i from instant T to $T+1$. We define the *actions vector* at time T as $\Pi[T] = (\Pi_1[T], \ldots, \Pi_\ell[T])$. Finally, $S_i[T]$ denotes the set of beliefs for context C_i at instant T whereas, accordingly, $\mathbf{S}[T]$ denotes the belief state $(S_1[T], \ldots, S_\ell[T])$. $\mathbf{S}[T]$ can be indicated simply as $\mathbf{S}$ whenever T does not matter.

Definition 6 (timed context) Let $C_i = \langle c_i, L_i, kb_i, br_i, OP_i, mng_i \rangle$ be a context in an mMCS. The corresponding *timed* context w.r.t. an initial belief state $\mathbf{S}$ is defined as:

- $C_i[0] = \langle c_i, L_i, kb_i[0], br_i, OP_i, mng_i, act_i, \mathscr{U}_i \rangle$
- $C_i[T+1] = \langle c_i, L_i, kb_i[T+1], br_i, OP_i, mng_i, act_i, \mathscr{U}_i \rangle$, for $T \geq 0$

where

- $kb_i[0] := kb_i$ and $\mathbf{S}[0] := \mathbf{S}$,
- $kb_i[T+1] := mng_i(app(\mathbf{S}[T]), kb')$ and $kb' \in \mathscr{U}_i(kb_i[T], \Pi_i[T])$,

with, for $T \geq 0$, $\mathbf{S}[T+1] = (S_1[T+1], \ldots, S_\ell[T+1])$, and, for each $i = 1, \ldots, \ell$, $S_i[T+1] \in ACC_i(mng_i(app(\mathbf{S}[T]), kb_i[T]))$.

Definition 7 (timed mMCS and timed data state) We let a *tmMCS* at time T be $M[T] = \{C_1[T], \ldots, C_\ell[T]\}$ and its *timed data state* is $\mathbf{S}[T] = (S_1[T], \ldots, S_\ell[T])$ as defined above.

The initial timed belief state $\mathbf{S}[0]$ can be an equilibrium, according to original mMCS definition. Later on, however, the transition from a timed belief state to the next one, and consequently the definition of an equilibrium, is determined both by the update operators and by the application of bridge rules. Therefore we have:

Definition 8 (timed equilibrium) A timed belief state of tmMCS M at time $T+1$ is a *timed equilibrium* whenever, for $1 \leq i \leq \ell$, it holds that $S_i[T+1] \in ACC_i(mng_i(app(\mathbf{S}[T+1]), kb_i[T+1]))$.

The meaning is that a timed equilibrium is now a data state which encompasses bridge rules applicability on the updated contexts' knowledge bases. As seen in Definition 6, applicability is checked on belief state $\mathbf{S}[T]$, but bridge rules are applied (and their results incorporated by the management function) on the knowledge base resulting from the update. The enhancement w.r.t. [12] is that the management function keeps its original role concerning bridge rules, while the update operator copes with updates, however and wherever performed. So, we relax the limitation that each rule involving an update should be a bridge rule, and that updates should consist of (the combination and elaboration of) simple atoms occurring in bridge bodies. Our approach, indeed, allows update operators to consider and incorporate any piece of knowledge; for instance, an update can encompass table creation and removal in a context which is a relational database or addition of machine-learning results in a context which collects and processes big data. In addition, we make time explicit thus showing the timed evolution of contexts and equilibria, while in [12] such an evolution is implicit in the notion of system run.

A very important recent proposal to introducing time in mMCS is that of Streaming MCS (sMCS) [25]. The sMCS approach equips MCS with data streams and models the time needed to transfer data among contexts, and computation time at contexts. The aim is to model the asynchronous behavior that may arises in realistic MCS applications. In sMCS, bridge rules can employ *window atoms* to obtain 'snapshots' of input streams from other contexts. Semantically, "feedback equilibria" extend the notion of MCS equilibria to the asynchronous setting, allowing local stability in system "runs" to be enforced, overcoming potentially infinite loops. To define window atoms the authors exploit the LARS framework [5], which allows to define "window functions" to specify *time-based windows (of size k)* out of a data stream S. A *Window atom* α are expressed on a plain atom A and can specify that A is true at some time instant or sometimes/always within a time window. Window atoms may appear in bridge rules: a literal $c_i : \alpha$ in the body of some bridge rule, where α is a window atom involving atom A, means that the "destination" context (the context to which the bridge rule belongs) is enabled to *observe A* in context c_i during given time window. This models in a natural way the access to sensors, i.e., cases where c_i is a fictitious context representing a sensor, whose outcomes are by definition observable. However, window atoms may concern other kinds of contexts, which are available to grant observability of their belief state. The sMCS approach also consider a very important aspect, namely that knowledge exchange between context is not immediate (as in the basic, rather ideal, MCS definition) but instead takes a time, that is supposed to be bounded, where bounds associated to the different contexts are known in advance. So, the approach considers the time that a context will take to evaluate bridge rules and to compute the management function. Thus, 'runs' of an sMCS will consider that each context's knowledge base will actually be updated according to these time estimations. The sMCS proposal is very relevant and orthogonal to ours. We in fact intend as future work to devise an integration of sMCS with our approach.

In the next section we consider aspects not presently covered in sMCS, i.e: how to activate bridge rule evaluation only when needed, and how to cope with the case where the time taken by knowledge interchange between contexts is not known in advance.

6 Static Set of Bridge Rules: Bridge-Rules Patterns

In the original MCS definition, each context is equipped with a static set of bridge rules, a choice that in tmMCSs can be a limitation. As mentioned, in fact, contexts to be queried are not necessarily fully known in advance; rather, it can become known only at "run-time" which are the specific contexts to be queried in the situations that practically arise. For instance, in [21] we have proposed an Agent Computational Environment (ACE), that is actually a tmMCS including an agent, to model a personal assistant-agent assisting a prospective college student; the student has to understand to which universities she could send an application (given the subjects of interest, the available budget for applications and then for tuition, and other preferences), where to perform the tests, where to enroll among the universities that have accepted the application. So, universities, test centers etc. have to be retrieved dynamically (as external contexts) and then suitable bridge rules to interact with such contexts must be generated and executed; in our approach, such bridge rules are obtained exactly by instantiating bridge rules patterns. See Section 7 below for a generalization of mMCS and tmMCS to dynamic systems, whose definition in term of composing contexts can change in time.

Thus, below we generalize and formalize for mMCS and tmMCS the enhanced form of bridge rules proposed in [22] for logical agents. In particular, we replace context names in bridge-rule bodies with special terms, called *context designators*, which are intended to denote a specific *kind* of context.

Example 7 Consider a literal such as $doctorRoss\!:\!prescription(disease,P)$ asking family doctor, specifically Dr. Ross, for a prescription related to a certain disease, might become $family\_doctor(d)\!:\!prescription(disease,P)$ where the doctor to be consulted is not explicitly specified. Each context designator must be substituted by a context name prior to bridge-rule activation; in the example, a doctor's name to be associated and substituted to the context designators $family\_doctor(d)$ (which, therefore, acts as a placeholder) must be dynamically identified; the advantage is, for instance, to be able to flexibly consult the physician who is on duty at the actual time of the inquiry.   □

In what follows we formalize what intuitively described in the previous example.

Definition 9 Let C_i be a context of a (t)mMCS. A *context designator* $m(k)$ is a term where m and k are symbols of the signature of L_i not occurring in kb_i or in br_i.

A *bridge-rule pattern* ϕ is an expression of the form:

$$s \leftarrow (\mathscr{C}_1\!:\!p_1),\ldots,(\mathscr{C}_j\!:\!p_j),not\,(\mathscr{C}_{j+1}\!:\!p_{j+1}),\ldots,not\,(\mathscr{C}_m\!:\!p_m)$$

where each $\mathscr{C}_d$ can be either a constant or a context designator.

New bridge rules can thus be obtained and applied by replacing, in a bridge-rule pattern, context designators via actual contexts names. So, contexts will now evolve also in the sense that they may increase their set of bridge rules by exploiting bridge-rule patterns:

Definition 10 Given a (t)mMCS, each of the timed composing contexts C_i, $1 \le i \le \ell$, is defined, at time 0, as $C_i[0] = \langle c_i, L_i, kb_i[0], br_i, brp_i, OP_i, mng_i, act_i, \mathscr{U}_i \rangle$, where brp_i is a set of bridge-rule patterns, and all the other elements are as in Definition 6.

Definition 11 (rule instance) An *instance* of the bridge-rule pattern $\phi \in brp_i$, for $1 \le i \le \ell$, occurring in a (t)mMCS, is a bridge rule r obtained by substituting every context designator occurring in ϕ by a context name $c \in \{c_1, \ldots, c_\ell\}$.

The context names to replace a context designator must be established by suitable reasoning in context's knowledge base, this by providing a suitable definition of a distinguished predicate $subs_i$ that we assume to occur in the signature of L_i.

Definition 12 Given a (t)mMCS M, any composing context C_i, for $1 \le i \le \ell$, and a timed data state $\mathbf{S}[T]$ of M, a *valid instance* of a bridge-rule pattern $\phi \in brp_i$ is a bridge rule $\hat{r}$ obtained by substituting every context designator $m(k)$ in ϕ with a context name $c \in \{c_1, \ldots, c_\ell\}$ such that $S_i[T] \models subs_i(m(k), c)$.

Let $vinst(brp_i)[T]$ denote the set of valid instances of bridge rule patterns that can be obtained at time T (for C_i). The set of bridge rules associated with a context increases in time with the addition of new ones which are obtained as valid instances of bridge-rule patterns.

Definition 13 Given a tmMCS, each of the timed composing contexts C_i, $1 \le i \le \ell$, is defined, at time $T+1$, as:

$$C_i[T+1] = \langle c_i, L_i, kb_i[T+1], br_i[T+1], brp_i, OP_i, mng_i, act_i, \mathscr{U}_i \rangle$$

with $br_i[T+1] = br_i[T] \cup vinst(brp_i)[T]$, $br_i[0] = br_i$, and all the rest is defined as done in Definition 6.

All the other previously-introduced notions (namely, equilibria, bridge-rule applicability, etc.) remain unchanged. That instantiation of bridge-rule patterns corresponds to specializing bridge rules with respect to the context which is deemed more suitable for acquiring some specific information at a certain stage of a context's operation. This evaluation is performed via the predicate $subs_i(m(k), c)$ that can be defined so as to take several factors into account, among which, for instance, trust and preferences.

A solution that might seem alternative though equivalent to ours could be that of defining "classical" bridge rules where each rule has a special element in the body acting as a 'guard' to establish if the rule should be actually applied. However, this alternative approach might reveal inappropriate. Let us reconsider our working examples. Initially, the student does not know which are the universities of interest, as this will be the result of a reasoning process and of knowledge exchange with other contexts; the student may even not know which universities exist. So, in the alternative solution it would be necessary to define one bridge rule for each university in

the world, where the rule has a guard to 'enable' the rule in case it should be finally applied as the student concluded to be interested: this is unfeasible if the student does not know all universities in advance, and it is clearly unpractical. A similar argument applies to our second example: doctors can retire or die, or start a new practice, or be initially unknown to the patient; so, they can hardly be all listed in advance. The same holds for many other classes of knowledge sources that can be modeled as contexts. Our solution is in many cases more practical and compact. It is unprecedented in the literature and adds actual expressive power.

This enhancement enables us to pursue the direction of *dynamic* (t)mMCS, where contexts can either join or leave the system during its operation. Such an extension has been in fact advocated since [10]. The meaning is that the set of contexts which compose an mMCS may be different at different times. Here, one can see the usefulness of having a constant acting as the context name; in fact, bridge-rule definition does not strictly depend upon the composition of the mMCS, as it would be by using integer numbers. Rather, the applicability of a bridge rule depends on the presence in the system of contexts with the names indicated in the bridge-rule definition. Not only can contexts be added or removed, but they can be substituted by new "versions" with the same name.

7 Static System Assumption and Unique Source Assumption: Dynamic (Timed) mMCSs (dtmMCS) and Multi-Source Option

As mentioned, in general, a heterogeneous collection of distributed sources will not necessarily remain static in time. New contexts can be added to the system, or can be removed, or can be momentarily unavailable due to network problems. Moreover, a context may be known by the others only via the role(s) it assumes or the services which it provides within the system. Although not explicitly specified in the original MCS definition, context *names* occurring in bridge-rule bodies must represent all the necessary information for reaching and querying a context, e.g., names might be URIs.

It is however useful for a context to be able to refer to other contexts via their roles, without necessarily being explicitly aware of their names. Also, a context which joins an MCS will not necessarily make itself visible to every other contexts: rather, there might be specific authorizations involved. These aspects may be modeled by means of the following extensions.

Definition 14 A *dynamic* managed (timed) Multi-Context System (d(t)mMCS) at time T is a (t)mMCS augmented with the two special contexts *Dir* and *Reach*, which have no associated bridge rules, bridge-rule patterns, and update operator, and where:

– *Dir* is a *directory* which contains the list of the contexts, namely $C_1, \ldots, C_\ell$, participating in the system at time T where, for each C_i, its name is associated with its *roles*. We assume *Dir* to admit queries of the form '*role@Dir*', returning the name of some context with role '*role*', where '*role*' is assumed to be a constant.
– *Reach* contains a directed graph determining which other contexts are *reachable* from each context C_i. For simplicity, we may see *Reach* as composed of couples

of the form (C_r, C_s) meaning that context C_s is (directly or indirectly) reachable from context C_r.

A d(t)mMCS is a more structured system w.r.t. MCS. Via *Dir*, contexts can access sources of knowledge without knowing them in advance, via a centralized system facility listing available contexts, with their role. In practice, contexts joining a d(t)mMCS will be allowed to register to *Dir* specifying their role, where this registration can be optional. Via *Reach*, the system assumes a structure that may correspond to the nature the applications, where a context can sometimes be not allowed to access all the others, either for a matter of security or for a matter of convenience. Often, information must be obtained via suitable mediation, while access to every information source is not necessarily either allowed or desirable. For instance, a patient's relatives can ask the doctors about the patient's conditions and prognosis, but they are not allowed (unless explicitly authorized by the patient) to access medical documentation directly. Students can see their data and records, but not those of other students.

For now, let us assume that a query $role@Dir = c$ where $c \in \{C_1, \ldots, C_\ell\}$, i.e., returns a unique result. The definition of timed data state remains unchanged. Bridge rule syntax must instead be extended accordingly:

Definition 15 Given a d(t)mMCS (at time T if timed) $M[T]$, each (non-ground) bridge rule r in the composing contexts has the form:

$$s \leftarrow (\mathscr{C}_1 : p_1), \ldots, (\mathscr{C}_j : p_j), not\,(\mathscr{C}_{j+1} : p_{j+1}), \ldots, not\,(\mathscr{C}_m : p_m).$$

where for $1 \leq k \leq m$ the expression $\mathscr{C}_k$ is either a context name, or an expression $role_k@Dir$.

Bridge-rule grounding and applicability must also be revised. In fact, for checking bridge rule applicability: (i) each expression $role_k@Dir$ must be substituted by its result and (ii) every context occurring in bridge rule body must be reachable from the context where the bridge rule occurs.

Definition 16 Let M be a d(t)mMCS (at time T if timed) and $\mathbf{S}$ be a (timed) data state for M. Let r be a bridge rule in the form specified in Definition 15. The *pre-ground* version r' of r is obtained by substituting each expression $role_k@Dir$ occurring in the body of r with its result c_k obtained from *Dir*.

Notice that r' is a bridge rule in "standard" form, and that r and r' have the same head, where their body differ since in r' all context names are specified explicitly.

Definition 17 Let r' be a pre-ground version of a bridge rule r occurring in context $\hat{C}$ of d(t)mMCS M with (timed) data state $\mathbf{S}$. Let ρ be a ground instance w.r.t. $\mathbf{S}$ of r'. We have now $hd(\rho) \in app(\mathbf{S})$ if ρ fulfills the conditions for applicability w.r.t. $\mathbf{S}$ and, in addition, for each context $\tilde{C}$ occurring in the body of ρ we have that $(\hat{C}, \tilde{C}) \in Reach$.

The definition of equilibria is basically unchanged, save the extended bridge-rule applicability. However, suitable update operators (that we do not discuss here) will be defined for both *Dir* and *Reach*, to keep both the directory and the reachability graph up-to-date with respect to the actual system state. The question may arise of

where such updates might come from. This will in general depend upon the application at hand: the contexts might themselves generate an update when joining/leaving a system, or some kind of monitor (that might be one of the composing contexts, presumably however equipped with reactive, proactive and reasoning capabilities) might take care of such task. Thus, the system as a whole will have a "policy" that defines reachability. Notice, in fact, that *Reach* provides structure to the system in a global way: each context indeed belongs to the sub-MCS of its reachable contexts. This corresponds to the structure of many applications and introduces a notion similar to "views" in databases: a context can sometimes be not allowed to access all the others, either for a matter of security or for a matter of convenience. Often, information must be obtained via suitable mediation, while access to every information source is not necessarily allowed or desirable. Reachability can evolve in time according to system's global policies.

Via *Dir* contexts are categorized, independently of their names, into, e.g., universities, doctors, etc. So, a context can query the right ones via their role. However, there might sometimes be the case where a specific context is not able to return a required answer, while another context with the same role instead would. More generally, we may admit a query *role@Dir* to return not just one, but possibly several results, representing the set of contexts which, in the given d(t)mMCS, have the specified role. So, the extension that we propose in what follows can be called a *multi-source option*. In particular, for d(t)mMCS $M[T]$, composed at time T of contexts $C_1, \ldots, C_\ell$, the expression $role_k@Dir$ occurring in bridge rule $r \in br_s$ will now denote some nonempty set $SC_k \subseteq (\{C_1, \ldots, C_\ell\} \setminus \{C_s\})$, indicating the contexts with the required role (where C_s is excluded as a context would not in this case intend to query itself). Technically, there will be now several pre-ground versions of a bridge rule, which differ relative to the contexts occurring in their body.

Definition 18 Let M be a d(t)mMCS (at time T) and $\mathbf{S}$ be a (timed) data state for M. Let $r \in br_s$ be a bridge rule in the form specified in Definition 15 occurring in context C_s. A *pre-ground* version r' of r is obtained by substituting each expression $role_k@Dir$ occurring in the body of r with $c \in SC_k$.

Bridge-rule applicability is still as specified in Definition 17 and the definition of equilibria is also basically unchanged.

In practice, one may consider to implement the multi-source option in bridge-rule run-time application by choosing an order for querying the contexts with a certain role as returned by the directory. The evaluation would proceed to the next one in case the answer is not returned within a time-out, or if the answer is under some respect unsatisfactory (according to the management function).

A further refinement might consist in considering, among the contexts returned by *role@Dir*, only the *preferred* ones. For instance, among medical specialist a patient might prefer the ones who practice nearer to patient's home, or who have the best ratings according to other patients' feedback.

Definition 19 (preferred source selection) Given a query *role@Dir* with result *SC*, a preference criterion $\mathscr{P}$ returns a (nonempty) ordered subset $SC^{\mathscr{P}} \subseteq SC$.

Different preference criteria can be defined according to several factors such as trust, reliability, fast answer, and others. To this aim we assume a predicate

$$prfs_i(role@Dir, SC, SCP)$$

to be defined in each context's knowledge base in order to select a subset SCP of a collection of contexts SC provided by the query $role@Dir$.

Example 8 In this example, the patient intends to take an appointment with a cardiologist, but she has no prior information on whom to consult, and she has no medical directory available. So, she asks the general directory about cardiologists. The directory will in general return a list of names. The second rule defines a preference criterion to filter the elements of such list. The predicate $closest(Site, Sites, Closest)$ determines, among the sites in the list $Sites$, the one which is closest to $Site$. This is done by exploiting $distance(Loc1, Loc2, Dist)$ that specifies the distance among two locations. Hence, the cardiologist is chosen whose medical clinic is the closest to patient's home. Then, the patient will take an appointment with the selected cardiologist C on day D at time T. (Here we exploited an aggregate to evaluate the minimum among a set of values, and adopted a syntax reminiscent of that of Answer Set Programming [9].)

$take\_cardiological\_visit\_appointment(C, D, T) \leftarrow$
            $cardiologist@Dir : take\_appointment(C, D, T).$

$prfs(cardiologist@Dir, CS, [Closest]) \leftarrow closest(home, CS, Closest).$

$closest(Site, Locs, Closest) \leftarrow$
            $distance(Site, Closest, Dist),$
            $Dist = \#min\{X : distance(Loc, C', X), member(L, Locs)\}.$
$distance(Site1, Site2, Dist) \leftarrow \ldots$

$\square$

Different existing approaches to preferences in logic programming might be applied to the present setting: see, among many, [12] and the references therein, [6,13] and [18–20]. The definition of a context will now be as follows.

Definition 20 A context C_i included in a d(t)mMCS (except for *Dir* and *Reach*) is defined (at time T), as $C_i[T] = \langle c_i, L_i, kb_i[T], br_i[T], brp_i, OP_i, mng_i, act_i, \mathcal{U}_i, \mathcal{P}_i \rangle$ where all elements are as defined before for (t)mMCS, and $\mathcal{P}_i$ is a preference criterion as specified in Definition 19.

Preferences have been exploited in MCS extensions in [28] and [38] in a very different way with respect to what is done here, to cope with relevant issue. Both the above-mentioned approaches are however orthogonal to ours, where the preference criterion is associated to each context, and determines which other contexts to query given a present situation. [38] aims to reconcile the different contexts' preferences about some common issue (for instance, mentioning from their first example where they apparently consider contexts as agents, whether to drink red or white wine if going to restaurant together). [28] copes with the situation where some constraint is

violated in some context depending upon other contexts' results communicated via bridge rules.

More precisely, [28] observes that "As the contexts of an MCS are typically autonomous and host knowledge bases that are inherited legacy systems, it may happen that the information exchange leads to unforeseen conclusions and in particular to inconsistency; to anticipate and handle all such situations at design time is difficult if not impossible, especially if sufficient details about the knowledge bases are lacking. Inconsistency of an MCS means that it has no model", i.e., no equilibrium. To solve the problem, they propose to define consistency-restoring rules based upon user preferences, specified in any suitable formalism (though in their examples they adopt CP-Nets [7]). This in order to choose among possible repairs, called "diagnoses", intended as modifications to bridge rules that might restore consistency. Such rules might be included into a special context to be added to given MCS. The goal of [38] instead is "to allow each agent (context) to express its preferences and to provide an acceptable semantics for MCS with preferences". To this aim, to define contexts they adopt "Ranked logics", where a partial order among acceptable belief sets is defined.

8 Logical Omniscience Assumption and Bridge Rules Application Mechanisms

In an implemented mMCS, as remarked in [4], "...computing equilibria and answering queries on top is not a viable solution." So, they assume a given MCS to admit an equilibrium, and define a query-answering procedure based upon some syntactic restriction on bridge-rule form, and involving the application and a concept of "unfolding" of positive atoms in bridge-rule bodies w.r.t. their definition in the "destination" context. Still, they assume an open system, where every context's contents are visible to others (save some possible restrictions). We assume instead contexts to be *opaque*, i.e., that contexts' contents are accessible from the outside only via queries.

Realistically therefore, the grounding of literals in bridge rule bodies w.r.t. the present data state will most presumably be performed at run-time, whenever a bridge rule is actually applied. Such grounding, and thus the bridge-rule result, can be obtained by "executing" or "invoking" literals in the body (i.e., querying contexts) left-to-right. In practice, we allow bridge rules to have negative literals in their body. To this aim, we introduce a syntactic limitation in the form of non-ground bridge rules very common in logic programming approaches. Namely, we assume that *(i) every variable occurring in the head of a non-ground bridge rule also occurs in some positive literal of its body; and (ii) in the body of each rule, positive literals occur (in a left-to-right order) before negative literals*.

So, at run-time variables in a bridge rule will be incrementally and coherently instantiated via results returned by contexts. Clearly, asynchronous application of bridge rules determines evolving equilibria.

As mentioned, we believe that bridge-rule application should not necessarily be reactive but rather, according to a context's own logic, other modalities of application may exist. For instance, a doctor will be consulted not whenever the doctor is available, but only if the patient is in need. Or, an application to some institution is

sent not just when all the needed documentation is available, but only if and when the potential applicant should choose to do that. Thus, in our approach the bridge rules that consult the doctor or issue the application remain in general "passive", unless they are explicitly triggered by suitable conditions. Moreover, the results returned by a bridge rule are not necessarily processed immediately, but only if and when the receiving context will have the wish and the need to take such results into account. For instance, the outcome of medical analyses will be considered only when a specialist is available to examine them. So, in our approach we make bridge-rule application proactive (i.e., performed upon specific conditions) and we detach bridge-rule application and the processing of the management function.

In the rest of this section we formalize the aspects that we have just discussed. Below, when referring to tmMCS we implicitly possibly refer to their timed dynamic version according to the definitions introduced in previous section. We are now able to formalize the rule instantiation and application mechanism, via a suitable notion of *potential applicability*.

A slight variation of the definition of bridge-rule grounding is required, as a generalization of Definition 1.

Definition 21 Let $r \in br_i$ be a non-ground bridge rule in a context C_i of a given tmMCS M with (timed) belief state $\mathbf{S}[T]$. A ground instance ρ of r w.r.t. $\mathbf{S}[T]$ is obtained by substituting every variable occurring in r with ground terms occurring in $\mathbf{S}[T]$.

By a "ground bridge rule" we implicitly mean a ground instance of a bridge rule w.r.t. a timed data state. We now redefine bridge-rule applicability, by first introducing proactive activation. For each context C_i, let

$$H_i[T] = \{s| \text{ there exists rule } \rho \in br_i \text{ with head } hd(\rho) = s \text{ at time } T\}. \quad (3)$$

We introduce a *timed triggering function*, $tr_i[T] : KB_i \rightarrow 2^{H_i[T]}$, which specifies (via their heads) which bridge rules are triggered at time T, and so, by performing some reasoning over the present knowledge base contents. Since $H_i[T]$ can be in general an infinite set, $tr_i[T](\cdot)$ may itself return an infinite set. However, in practical cases, it will presumably return a finite set of rules to be applied at time T.

Definition 22 A rule $\rho \in br_i$ is *triggered at time T* iff $hd(\rho) \in tr_i[T](kb_i[T])$.

Example 9 The following rules specify a simple bridge rule and its triggering condition. Hence, the first rule is activated whenever the body of the second rules is satisfied. Namely, one is allowed to proceed in finding a clinic only if there is a diagnosis indicating that an operation is actually needed.

$find\_clinic(H,P,D) \leftarrow$
  $patient(P), operation(H), disease(D), ...$
$tr(find\_clinic(H,P,D)) \leftarrow$
  $diagnosis(medical\_specialist(M),patient(P),disease(D),needs(operation(H)))$.
  □

Let $grtr_i[T]$ be the set of all ground instances of bridge rules which have been triggered at time T, i.e., $\rho \in grtr_i[T]$ iff ρ is a ground instance w.r.t. $\mathbf{S}[T]$ of some non-ground r such that $hd(\rho) \in tr_i[T](kb_i[T])$.

For a tmMCS M, data state $\mathbf{S}[T]$ and a ground bridge rule ρ, let $\mathbf{S}[T] \models body(\rho)$ represent that the rule body holds in belief state $\mathbf{S}[T]$.

Definition 23 The set $app(\mathbf{S}[T])$ relative to ground bridge rules which are applicable in a timed data state $\mathbf{S}[T]$ of a given tmMCS $M[T] = \{C_1[T], \ldots, C_\ell[T]\}$ is defined as:

$$app(\mathbf{S}[T]) = \{hd(\rho) \mid \exists T' \le T \text{ such that } \rho \in grtr_i[T'] \text{ and } \mathbf{S}[T] \models body(\rho)\}.$$

By the definition of $app(\cdot)$, a (ground instance of) a bridge rule can be triggered at a time T', but can then become applicable at some later time $T \ge T'$. Thus, any bridge rule which has been triggered remains in predicate for applicability, which will occur whenever its body is entailed by some future data state. One alternative solution that may seem equivalent to the proposed one is that of adding a "guard" additional positive subgoal in a bridge-rule body, to be proved within the context itself. Such additional literal would have the role of enabling bridge-rule application. However, in our proposal a context is *committed* to actually apply bridge rules which have been triggered as soon as they will (possibly) be applicable while an additional subgoal should be re-tried at each stage and, if non-trivial reasoning is involved, this may be unnecessary costly.

The definition of timed equilibria remains unchanged, apart from the modified bridge-rule applicability. However, the added (practical) expressiveness is remarkable as a context in the new formulation is not just the passive recipient of new information, but it can reason about which bridge rules to potentially apply at each stage.

As mentioned, for practical applications it may be useful to allow the grounding of literals in bridge-rule bodies to be computed at run-time, whenever a bridge rule is actually applied. Because of the adoption of a left-to-right execution and in virtue of the assumptions (i)-(ii) seen earlier, the grounding of a bridge-rule is obtained by "invoking" literals in its body so as to obtain the grounding of positive literals first, to be extended later to negative ones.

Each positive literal $(c_j : p)$ in the body of a bridge rule may fail (i.e., c_j will return a negative answer), if none of the instances of p given the partial instantiation computed so far is entailed by c_j's present data state. Otherwise, the literal succeeds and the other ones are (possibly) instantiated accordingly. Negative literals $not\,(c_j : p)$ make sense only if p is ground at the time of invocation, and succeed if p is *not* entailed by c_j's present data state. In case either some literal fails or a non-ground negative literal is found, the overall bridge rule evaluation fails without returning results. Otherwise, the evaluation succeeds, and the result can be elaborated by the management function of the "destination" context.

For modeling such a run-time procedural behavior, we introduce *potential applicability*. Note that, since literals in bridge-rule bodies are ordered left to right as they appear in the rule definition, we can refer to the first, second, etc., positive/negative literal.

Definition 24 A ground bridge rule $\rho \in grtr_i[T']$ for some T', of the form (1) with h positive literals $A_1, \ldots, A_h$ in its body is *potentially applicable* to grade $k \leq h$ at time $T \geq T'$ iff given a reduced version ρ' of the rule of the form $s \leftarrow A_1, \ldots, A_k$ we have $\mathbf{S}[T] \models body(\rho')$.

Hence, a triggered bridge rule ρ is potentially applicable to grade k at time T if the timed belief state at time T entails its first k positive literals. Plainly, we have:

Lemma 1 *A ground bridge rule $\rho \in grtr_i[T']$ for some T', of the form (1) with h positive literals $A_1, \ldots, A_h$ in its body which is potentially applicable to grade $k \leq h$ at time T is also potentially applicable at time T to grade k', $1 \leq k' < k$.*

Proof This comes from the fact that if $\mathbf{S}[T] \models body(\rho')$ with ρ' of the form $s \leftarrow A_1, \ldots, A_k$, then it will hold that $\mathbf{S}[T] \models body(\rho'')$ with ρ'' of the form $s \leftarrow A_1, \ldots, A_k'$. This because entailing a conjunction implies entailing any sub-conjunction. □

Proposition 4 *A ground bridge rule ρ is* potentially applicable *to grade k at time T_k iff for every k' such that $1 \leq k' \leq k$ there exists $T_{k'} \leq T_k$ such that ρ is potentially applicable to grade k' at time $T_{k'}$.*

Proof This is a plain consequence of the above lemma if $T_{k'} = T_k$. □

It will most often be the case that a bridge-rule body will become potentially applicable to a higher and a higher grade as time passes. I.e., often for some k' it will be $T_{k'} < T_k$.

Proposition 5 *Let $\rho \in grtr_i[T']$ be a ground bridge rule, for some T', of the form (1) which is potentially applicable to grade h (or simply "potentially applicable") at time T; then, $hd(\rho) \in app(\mathbf{S}[T])$ iff for every negative literal $not\,(c_k{:}p)$ in the body, $h+1 \leq k \leq m$, $p \notin S_k[T]$.*

Proof The result is obtained trivially from the definition of bridge-rule applicability in a data state, here $\mathbf{S}[T]$, which requires the positive body to be entailed and the negative body not to be entailed. □

As the contexts composing a tmMCS may be non-monotonic, a bridge rule ρ can become potentially applicable at some time and remain so without being applicable, but can become applicable at a later time if the atoms occurring in negative literals are no longer entailed by the belief state at that time.

Notice that, in fact, in a practical distributed setting, literals in the body may succeed/fail at different times, depending on the various context update times, and upon network delay. Let us therefore assume that the success of a positive literal A (resp. negative literal $not\,A$) is annotated with the time-stamp when such success occurs, of the form $A[T]$ (resp. of the form $not\,A[T]$). So, for any bridge rule $\rho \in grtr_i[T']$ of the form (1) which is triggered at some time and then executed later, by the expression

$$s[T_s] \leftarrow A_1[T_1], \ldots, A_h[T_h], not\,A_{h+1}[T_{h+1}], \ldots, not\,A_m[T_m]$$

we mean that the rule has been executed left-to-right where each literal $A[T]/not\,A[T]$ has succeeded at time T and the result has been processed (via the management function) by the destination context (i.e., the one where the bridge rule occurs) at time T_s. Clearly, we have $T_1 \leq T_2 \leq ... \leq T_m \leq T_s$. The above expression is called the *run-time version* of rule ρ. It is *successful* if all literals succeed; otherwise it is *failed*.

In many practical cases we are able to assume a certain degree of persistence in the system, i.e., that a literal which succeeds at a time T_r would then still succeed (if re-invoked) at every subsequent time T_q with $T_r \leq T_q \leq T_s$ ("local persistence assumption" for rule ρ). This corresponds to assuming that during the execution of a bridge-rule no changes in the involved contexts occur that would invalidate the bridge-rule result before actual completion; so, this would enforce what we call the "coherent" execution of a bridge rule. This assumption may sometimes be problematic in practice, but in a distributed system such as a tmMCS we cannot realistically assume the execution to be instantaneous. The assumption of persistence can be considered as reasonable whenever the time amount required by the execution of a bridge rule is less than the average system change rate with respect to the involved belief elements. This is the usual condition for artificial intelligence systems to be adequate w.r.t. the environment where they are situated, rather that being "brittle". There are possible ways of enforcing the assumption. For instance, the contexts involved in a bridge rule execution might commit to keep the involved belief elements untouched until the destination context sends an "ack" to signal the completion of rule execution. This or other strategies to ensure coherent execution are deferred to the implementation of tmMCS. Under this assumption we have:

Theorem 1 *Given a successful run-time version of a ground bridge rule $\rho \in grtr_i(\hat{T})$, for some $\hat{T}$, of the form $s[T_s] \leftarrow A_1[T_1], \ldots, A_h[T_h], not\,A_{h+1}[T_{h+1}], \ldots, not\,A_m[T_m]$. Then, $hd(\rho) \in app(\mathbf{S}[T_s])$ and there exists $T \leq T_m$ such that $hd(\rho) \in app(\mathbf{S}[T])$. Moreover, for each $i \leq j$ there exists $T_i' \leq T_i$ and $T_{i-1} \leq T_i'$ if $i > 1$ such that ρ is potentially applicable to grade i at time T_i'.*

Proof (Sketch). Thanks to the persistence of the system, putting $T_i' = T_i$, for each $i \leq j$, suffices to prove the first property, and putting $T = T_m$ suffices for the second one. The third one follows in a straightforward way. □

The relevance of the above theorem is because, in practice, the execution involves a non-ground rule $r \in tr_i[\hat{T}](kb_i[\hat{T}])$ whose rule ρ mentioned therein is a ground instance. Then, successful left-to-right execution of literals in the body of r will lead to dynamically generate at run-time a successful run-time version of ρ. Specifically, the execution of each non-ground positive literal in the body of r will generate a partial instantiation which is propagated to the rest of the bridge-rule body.

9 Case Study

In this section we provide an example to illustrate the new features. We refer to Example 2 in [12] where it is supposed that a user, Bob, suffering from dementia, is able to live at home thanks to an assisted living system, modeled by means of an rMCS.

The system is in fact able to constantly track Bob's position, and to detect and take care of emergencies (such as a sudden illness, but also Bob's forgetting the stove on, and the like), on the basis of the data provided by sensors suitably installed in Bob's flat.

We generalize the rMCS to an F&K which is able to detect changes in Bob's health and is capable to notice anomalous behavior which might signify an impaired state. The inputs to the system can be provided by means of sensors, and by recording and monitoring Bob's activities via suitable telemedicine appliances: by adopting ubiquitous sensing technologies, such appliances since long (cf., e.g., [35]) include several kinds of smart devices such as, e.g., beds that monitor sleep, chairs that monitor breathing and pulse, video cameras that monitor general wellbeing, and more. We assume that the overall F&K system can attend to several patients. Each patient, among which Bob, is in care of a personalized assistant/monitoring agent (PMA) participating in the F&K. In [35] it is in fact remarked that, without such an integrated system, sophisticated telemedicine appliances are "little more than emergency alarm systems". A PMA might be incarnated, in different use-cases, in a chatterbot/avatar on TV or on the mobile phone, in a smart device, or even in a social/care robot. In our view, a PMA should be able to "transmigrate" from one form to another one according to the situation which a patient is experimenting at a certain stage of her/his activities, so that a PMA will never desert "her" patient. The patient's PMA (in this case Bob's), is assumed below to be modeled as a context included in an mMCS where the PMA is in particular an agent defined in any agent-oriented logic language [16]. As said, this agent will be adequately situated and connected to the patient.

Bob's PMA pma_{bob} might for instance, in case of slight variations of, e.g., blood pressure or blood coagulation, re-adjust the quantity of medicament by consulting a treatment knowledge base. In case of more relevant symptoms, the agent might consult either Bob's physician or a specialist, depending upon the relative importance of symptoms. In case of an emergency, for instance a hemorrhage, immediate help is asked for. Clearly, the example is over-simplified, and we do not aim at medical accuracy.

Following [12], in the example we assume that present "objective" time is made available to each context as $now(T)$, where the current value of T is entered into the system by a particular sensor. As previously specified, literals in bridge-rule bodies with no indication of a context are to be proved locally to the context to which the bridge rule belongs.

The following bridge rule associated to each patient's PMA, and thus also to pma_{bob}, is maybe the most crucial as it is able to ask for immediate help. The last literal in the body concerns a context $emergency\_center$, which is supposed to be an interactive component for emergency management able to send, for instance, an ambulance or a helicopter for transportation to the hospital. The last literal in bridge-body will succeed whenever the $emergency\_center$ context has received and processed the request, which is sent in case severe symptoms have been detected. Those symptoms and the time when the request is issued are communicated to the emergency center together with the patient's data (here, for simplicity, just a patient's identification code). The bridge rule head is processed by the management function so as to simply

add to the PMA's knowledge base the record of the fact that help has been required at time T for set of severe symptoms S.

$help\_asked(bob,S,T,T1,Th,H) \leftarrow$
$\qquad now(T), detected\_symptoms(S,T1), T1 \leq T,$
$\qquad patientid(bob,Bid),\ emergency\_center : urgent\_help(Bid,S,T,Th,H)$

The emergency center is provided with Bob's patient's id Bid and returns the time Th and mean H by which the emergency will be coped with (e.g., an ambulance by 15 minutes). In order to ensure application of the rule only in case of a real emergency, it will be triggered (by adding its head to $tr_{pma_{bob}}[T]$) only upon specific conditions, e.g., if symptoms have occurred that can be considered to be severe for Bob's particular health conditions and a physician is not already present or promptly available. Upon bridge-rule application, the management function will record the (ground) request for help $help\_asked(bob,S,T,T1,Th,H)$ that has been issued; this allows for instance the PMA to check whether the promised assistance will actually arrive in time and possibly to minister palliative treatment in the meanwhile.

The following bridge rule, potentially crucial for cardiopathic patients, will be triggered by in case the blood coagulation value detected at time T is anomalous; this implies that the quantity of anti-coagulant which Bob takes to treat his heart disease must be rearranged accordingly. The correct quantity Q is obtained by the ATC (Anti-Coagulant Center) knowledge base according to the last blood coagulation value V and its variation D from previous records.

$quantity(anticoagulant,Q) \leftarrow$
$\qquad coagulation\_val(V,D), patientid(bob,Bid),\ atc : quantity(Bid,V,D,Q)$

In case a patient's health state is not really critical but is anyway altered, a physician must be consulted. However, in case, for example, of a simple flu the family doctor suffices, while if there are symptoms that might be related to a more serious condition then a specialist (e.g., a cardiologist) should be consulted. Thus, there will be a bridge-rule pattern of the following form, where a physician can be consulted for condition C (where C is either symptom or a list of symptoms). Again, the management function will record the request having been sent; the last literal in the body will succeed, when the rule is dynamically executed, as soon as the doctor receives the request.

$call\_physician(bob,T) \leftarrow$
$\qquad now(T), condition(bob,T,C),$
$\qquad patientid(bob,Bid),\ mydoctor(d) : consultation\_needed(Bid,C,T)$

The physician, represented by the context designator $mydoctor(d)$, should however be determined according to C. This can be done by suitably augmenting the definition of the distinguished predicate $subs_{pmi\_bob}$, for instance, as follows. The notation '_' indicates a "don't care" variable, as time is not taken into account here.

$subs_{pmi\_bob}(mydoctor(d),F) \leftarrow family\_doctor(F), condition(bob,\_,fever)$
$subs_{pmi\_bob}(mydoctor(d),F) \leftarrow family\_doctor(F), condition(bob,\_,headache)$
$subs_{pmi\_bob}(mydoctor(d),G) \leftarrow my\_cardiologist(G), condition(bob,\_,chestpain)$
$my\_neurologist(G), condition(bob,\_,[dizziness,swoon])$
$\ldots$

Thus, a valid instance of the bridge-rule pattern will be generated according to the patient's need, as evaluated by the patient's PMI. The resulting bridge rule will then be immediately executed. So for instance, if Bob has chest pain and the cardiologist who has been following him is Dr. House, then the bridge rule below will be constructed and triggered:

$call\_physician(bob, T) \leftarrow$
$\quad\quad now(T), condition(bob, T, chest\_pain),$
$\quad\quad patientid(bob, Bid), drHouse : consultation\_needed(Bid, chest\_pain, T)$

As a further generalization, the PMA might also try to retrieve a doctor (for instance, a dermatologist, assuming that Bob has never consulted one in recent times, and so there is no record in the PMA's knowledge base) via the system's directory; in this case $subs_{pmi\_bob}$, would include a rule such as:

$subs_{pmi\_bob}(mydoctor(d), F) \leftarrow dermatologist@Dir, condition(bob, \_, eczema)$

Suitable preferences might be specified, stating for instance that a dermatologist who is an expert in allergic problems is preferred (if any can be found), and in case more than one can be located, the doctor who has the medical office nearer to Bob's home is preferred. So we might state:

$\mathscr{P}_{pmi\_bob} = \{expert(dermatology, allergy), near\_home(BobAddress)\}$

Such specification presumes that the outcome provided by the Directory is suitably processed by means of a fragment of a program present in contexts' specification, in line with what shown in Example 8.

10 Complexity

We briefly discuss the complexity issues related to the proposed approach. In general, the property that we may wish to check is whether a specific belief of our interest will eventually occur at some stage in one (or all) timed equilibria of a given mMCS. The formal definition is the following.

Definition 25 The problem $Q^\exists$ (respectively $Q^\forall$), consists in deciding whether, for a given mMCS M under a sequence $\Pi = \Pi[1], \Pi[2], \ldots, \Pi[t]$ of update actions performed at time instants $1, 2, \ldots, t$, and for a context C_i of M and a belief p_i for C_i, it holds that $p_i \in S_i[t']$ for some (respectively for all) timed equilibria $\mathbf{S}[t']$ at time $t' \leq t$.

We resort, like [12], to *context complexity* as introduced in [27]. One has first to consider a *projected belief state* $\hat{\mathbf{S}}[t]$, which includes in the element $\hat{S}_i[t]$ the belief b_i one wants to query, and also includes for every element $\hat{S}_j[t]$ the beliefs that contribute to bridge-rule applications which may affect p_i (see [4] for an algorithm which computes such sets of beliefs for rMCSs). Then, the context complexity for C_i is the complexity of establishing whether the element $\hat{S}_i[t]$ of such projected belief state is a subset of the corresponding element $\hat{S}_i[t]$ of some timed equilibrium at time t.

Definition 26 The system context complexity of M is a (smallest) upper bound for the context complexity classes of its component contexts.

Thus, the system context complexity depends upon the logics of the contexts in M. The problems $Q^\exists$ and $Q^\forall$ are however undecidable for infinite update sequences, because contexts' logics can in general simulate a Turing Machine and thus such problems reduce to the *halting* problem. Actual complexity results can however be obtained under some restrictions. In particular, we assume that all contexts C_i's knowledge bases and belief states are finite at any stage, all update functions $\mathscr{U}_i$, management functions mng_i and triggering functions tr_i are computable in polynomial time, and that the set of bridge-rule patterns is empty and all bridge rules are ground. then we can state the following.

Theorem 2 *For finite update sequences, the system context complexity determines the complexity of membership of $Q^\exists$ and, complementarily, the complexity of $Q^\forall$. Hardness holds if it holds for the system context complexity.*

Proof A projected belief state can be guessed for each stage $T \in \mathbb{N}$ by a non-deterministic Turing machine. Then, the inclusion of each such projected belief state $\hat{\mathbf{S}}[T]$ in some (all) timed equilibria $\mathbf{S}[T]$ at that stage can be established by an oracle under the system's context complexity and, if the answer is positive, it must be checked whether $p_i \in \hat{S}_i[T]$; if not, subsequent updates must be performed (in polynomial time) over $\hat{\mathbf{S}}[T]$, and the two checks must be repeated at each stage; this until either p_i is found or time T is reached, thus obtaining either a positive or a negative answer to the $Q^\exists$ problem.

The complexity of consistency checking for an mMCS M, i.e., the complexity of deciding whether it has some equilibrium, has been discussed in [10] and remains the same for many variants of (m)MCS, including our own, as it depends on context complexity. Some cases are shown in tables reported in [10, 12, 36].

Reconsidering bridge-rule patterns and non-ground bridge rules we observe that by assuming that $subs_i$ can be computed in polynomial time and produces a finite number of substitutions for each context designator, and given a set of bridge-rule patterns of a certain cardinality (size) $\hat{c}$, the size of the set of its valid instances is in general single exponential in $\hat{c}$. The same holds in principle for the grounding of bridge rules. However, in our setting we assume that bridge-rule are executed and grounded in a Prolog-like fashion: therefore, the computational burden for grounding is consequently smaller.

11 Concluding Remarks

In this paper we have proposed extensions to MCSs, aimed at practical application of such systems. We have outlined a number of aspects that could be improved, and we have then specified and formalized possible improvements. The proposed modifications are incremental (though they might be employed separately), and in fact each definition is presented as an extension of previous ones. Thus, the final result is an overall improved MCS formalization coping with all previously emphasized problematic aspects. Notice that, given a dtmMCS at time t, the system enjoys all the properties of the corresponding mMCS, i.e., the mMCS obtained by including

the same contexts (except *Dir* and *Reach*), each with the associated bridge rules (neglecting bridge rules patterns), and where each expression of the form *role@Dir* has been substituted by a context name according to *Dir* and *Reach*, and to the preferences.

The enhanced customizable bridge rules that we have formalized here have been successfully experimented (though in a preliminary embryonic version) in a significant case-study, reported in [15]. We intend to employ the proposed features in the implementation, that will start in the near future, of an F&K Cyber-Physical System and in the activities of the DIGFORASP COST Action on Digital Forensics. This project will allow us to perform practical experiments in order to assess the performance of this kind of systems (as in fact, as seen, not much can be said about complexity) and to identify possible limitations and/or further aspects that can be subjected to improvements.

References

1. Aielli, F., Ancona, D., Caianiello, P., Costantini, S., De Gasperis, G., Di Marco, A., Ferrando, A., Mascardi, V.: FRIENDLY & KIND with your health: Human-friendly knowledge-INtensive dynamic systems for the e-health domain. In: K. Hallenborg, S. Giroux (eds.) Proc. of A-HEALTH at PAAMS 2016., Comm. in Computer and Inf. Sci. Springer (2016)
2. Alferes, J.J., Brogi, A., Leite, J.A., Pereira, L.M.: Evolving logic programs. In: Logics in Artificial Intelligence, Proc. of the 8th Europ. Conf., JELIA 2002, LNAI 2424, pp. 50–61. Springer-Verlag, Berlin (2002)
3. Apt, K.R., Bol, R.N.: Logic programming and negation: A survey. The Journal of Logic Programming **19-20**, 9–71 (1994)
4. Barilaro, R., Fink, M., Ricca, F., Terracina, G.: Towards query answering in relational multi-context systems. In: P. Cabalar, T.C. Son (eds.) Proc. of LPNMR 2013, *LNCS*, vol. 8148, pp. 168–173. Springer (2013)
5. Beck, H., Dao-Tran, M., Eiter, T.: LARS: A logic-based framework for analytic reasoning over streams. Artif. Intell. **261**, 16–70 (2018)
6. Bienvenu, M., Lang, J., Wilson, N.: From preference logics to preference languages, and back. In: Proc. of the Twelfth Intl. Conf. on the Principles of Knowledge Repr. and Reasoning (KR 2010), pp. 414–424 (2010)
7. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. J. Artif. Intell. Res. **21**, 135–191 (2004)
8. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: Proc. of the 22nd AAAI Conf. on Artificial Intelligence, pp. 385–390. AAAI Press (2007)
9. Brewka, G., Eiter, T., (eds.), M.T.: Answer set programming: Special issue. AI Magazine **37**(3) (2016)
10. Brewka, G., Eiter, T., Fink, M.: Nonmonotonic multi-context systems. In: Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning, *LNCS*, vol. 6565, pp. 233–258. Springer (2011)
11. Brewka, G., Eiter, T., Fink, M., Weinzierl, A.: Managed multi-context systems. In: Proc. of IJCAI 2011, pp. 786–791. AAAI (2011)
12. Brewka, G., Ellmauthaler, S., Pührer, J.: Multi-context systems for reactive reasoning in dynamic environments. In: T. Schaub (ed.) Proc. of ECAI 2014. AAAI (2014)
13. Brewka, G., Niemelä, I., Truszczyński, M.: Preferences and nonmonotonic reasoning. AI Magazine **29**(4) (2008)
14. Cabalar, P., Costantini, S., Formisano, A.: Multi-context systems: Dynamics and evolution. In: B. Bogaerts, A. Harrison (eds.) Proceedings of the 10th Workshop on Answer Set Programming and Other Computing Paradigms co-located with the 14th International Conference on Logic Programming and Nonmonotonic Reasoning, ASPOCP@LPNMR 2017., *CEUR Workshop Proceedings*, vol. 1868. CEUR-WS.org (2017). URL `http://ceur-ws.org/Vol-1868`

15. Costantini, S.: ACE: a flexible environment for complex event processing in logical agents. In: EMAS 2015, Revised Selected Papers, *LNCS*, vol. 9318. Springer (2015)
16. Costantini, S.: Knowledge acquisition via non-monotonic reasoning in distributed heterogeneous environments. In: M. Truszczyński, G. Ianni, F. Calimeri (eds.) Proc. of LPNMR 2013, *LNCS*, vol. 9345. Springer (2015)
17. Costantini, S., De Gasperis, G.: Exchanging data and ontological definitions in multi-agent-contexts systems. In: A. Paschke, P. Fodor, A. Giurca, T. Kliegr (eds.) RuleMLChallenge track, Proceedings, CEUR Workshop Proceedings. CEUR-WS.org (2015)
18. Costantini, S., Formisano, A.: Modeling preferences and conditional preferences on resource consumption and production in ASP. Journal of Algorithms in Cognition, Informatics and Logic **64**(1) (2009)
19. Costantini, S., Formisano, A.: Augmenting weight constraints with complex preferences. In: Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, pp. –. AAAI Press, USA (2011)
20. Costantini, S., Formisano, A.: Weight constraints with preferences in ASP. In: Proc. of the 11th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR2011), Lecture Notes in Computer Science. Springer (2011)
21. Costantini, S., Formisano, A.: Augmenting agent computational environments with quantitative reasoning modules and customizable bridge rules. In: N. Osman, C. Sierra (eds.) Autonomous Agents and Multiagent Systems - AAMAS 2016 Workshops, - Visionary Papers, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 10003, pp. 104–121 (2016)
22. Costantini, S., Formisano, A.: Augmenting agent computational environments with quantitative reasoning modules and customizable bridge rules. In: M. Baldoni, J.P. Müller, I. Nunes, R. Zalila-Wenkstern (eds.) Proc. of EMAS 2016, *Lecture Notes in Computer Science*, vol. 10093, pp. 192–209. Springer (2016). Also in [21]
23. Costantini, S., Gasperis, G.D.: Bridge rules for reasoning in component-based heterogeneous environments. In: J.J. Alferes, L.E. Bertossi, G. Governatori, P. Fodor, D. Roman (eds.) Rule Technologies. Research, Tools, and Applications - 10th International Symposium, RuleML 2016. Proceedings, *Lecture Notes in Computer Science*, vol. 9718, pp. 97–112. Springer (2016)
24. Costantini, S., Gasperis, G.D., Olivieri, R.: Digital forensics evidence analysis: An answer set programming approach for generating investigation hypotheses. In: M.T. Francesco Calimeri Giovambattista Ianni (ed.) Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Proceedings, *Lecture Notes in Computer Science*, vol. 9345, pp. 228–241. Springer (2015). Long version in CEUR Workshop Proceedings of CILC 2015, 30th Italian Conference of Computational Logic
25. Dao-Tran, M., Eiter, T.: Streaming multi-context systems. In: C. Sierra (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 1000–1007. ijcai.org (2017)
26. Dao-Tran, M., Eiter, T., Fink, M., Krennwallner, T.: Distributed evaluation of nonmonotonic multi-context systems. J. of Artificial Intelligence Research **52**, 543–600 (2015)
27. Eiter, T., Fink, M., Schüller, P., Weinzierl, A.: Finding explanations of inconsistency in multi-context systems. In: F. Lin, U. Sattler, M. Truszczyński (eds.) Proc. of KR 2010. AAAI (2010)
28. Eiter, T., Weinzierl, A.: Preference-based inconsistency management in multi-context systems. J. Artif. Intell. Res. **60**, 347–424 (2017)
29. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming. AI Magazine **37**(3), 53–68 (2016)
30. Fink, M., Ghionna, L., Weinzierl, A.: Relational information exchange and aggregation in multi-context systems. In: Proc. of LPNMR 2011, *LNCS*, vol. 6645, pp. 120–133 (2011)
31. Gelfond, M.: Answer sets. In: Handbook of Knowledge Representation, chap. 7. Elsevier (2007)
32. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: R. Kowalski, K. Bowen (eds.) Proc. of the 5th Intl. Conf. and Symposium on Logic Programming (ICLP/SLP'88), pp. 1070–1080. The MIT Press (1988)
33. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9**, 365–385 (1991)
34. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics or: How we can do without modal logics. Artif. Intell. **65**(1), 29–70 (1994)
35. Glascock, A., Kutzik, D.: Essential Lessons for the Success of Telehomecare: Why It's Not Plug and Play. Assistive technology research series. IOS Press (2012)

36. Gonçalves, R., Knorr, M., Leite, J.: Evolving bridge rules in evolving multi-context systems. In: Proc. of CLIMA 2014, *LNCS*, vol. 8624, pp. 52–69. Springer (2014)
37. Gonçalves, R., Knorr, M., Leite, J.: Evolving multi-context systems. In: Proc. of ECAI 2014, pp. 375–380. IOS (2014)
38. Le, T., Son, T.C., Pontelli, E.: Multi-context systems with preferences. Fundam. Inform. **158**(1-3), 171–216 (2018)
39. Lloyd, J.W.: Foundations of Logic Programming. Springer-Verlag (1987)
40. Marek, V.W., Truszczyński, M.: Stable logic programming - An alternative logic programming paradigm. In: 25 years of Logic Programming Paradigm, pp. 375–398. Springer (1999)
41. Olivieri, R.: Digital forensics meets complexity theory and artificial intelligence: Towards automated generation of investigation hypothesis. Ph.D. thesis, Ph.D. Program in Information and Communication Technology, University of L'Aquila, Italy (2016)
42. Roelofsen, F., Serafini, L., Cimatti, A.: Many hands make light work: Localized satisfiability for multi-context systems. In: R. López de Mántaras, L. Saitta (eds.) Proc. of ECAI 2004, pp. 58–62. IOS Press (2004)