Multi-Context Systems: Dynamics and Evolution*

Pedro Cabalar[1], Stefania Costantini[2], and Andrea Formisano[3]

[1] University of Corunna, Spain, email: cabalar@udc.es
[2] Università di L'Aquila, Italy, email: stefania.costantini@univaq.it
[3] Università di Perugia, Italy, email: andrea.formisano@unipg.it

Abstract. Multi-Context Systems (MCS) model, using 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 application.

1 Introduction

Multi-Context Systems (MCSs) have been proposed in Artificial Intelligence and Knowledge Representation to model information exchange among heterogeneous sources [3, 4, 6]. 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 Datalog or Answer Set Programming (ASP) rules, save that atoms in their bodies refer to knowledge items to be obtained from external contexts. So, a literal $(c_j : p)$ (where p is an atom) in the body of a bridge rule occurring in context c_i means that p can be proved in context c_j (given c_j's underlying logic and semantics and given c_j's knowledge base present contents), vice versa $not\ (c_j : p)$ means p cannot be proved in c_j. In case the entire body "succeeds" then the bridge rule can be applied, and its conclusion can be exploited within context c_i.

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 involve agents; MCSs encompassing logical agents have in fact been proposed in [8].

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

* This work is partially supported by INdAM-GNCS-17 project, by Xunta de Galicia, Spain (projects GPC ED431B 2016/035 and 2016-2019 ED431G/01 for CITIC center) and by the European Regional Development Fund (ERDF).

paper considers the following aspects. (i) The specification of *proactive* bridge-rule activation, i.e., a bridge-rule in our approach is no longer applied whenever applicable, but it must be explicitly enabled by the context where it occurs. (ii) The explicit definition of *evolution* of an MCS over time in consequences of updates that can be more general than sensor input as treated in [6]. (iii) A specification of bridge-rule grounding in an evolving MCS. (iv) The definition of practical modalities of execution of bridge rules in a distributed MCS. (v) *Bridge rules patterns*, by which we make bridge rules parametric w.r.t. contexts by introducing special terms called *context designators* to be replaced by actual context names; this is in our opinion a very important extension, which allows general bridge rules schemata to be specialized to actual bridge rules by choosing which specific contexts (of a certain kind) should specifically be queried in the situation at hand. There is a relation with the work of [14], where, at run-time, new bridge rules can be "observed", i.e., learned from outside. Such rules can be added to previous ones, or can replace them with a new version, with a check on consistency among bridge-rules conclusions after the update. This work is complementary with our proposal, where new bridge rules are generated inside a context.

Overall, we propose enhancements which are not in conflict with existing proposals for MCSs which evolve in time [6, 15, 14], but rather either generalize such proposals or, at least, can be combined with them. In what follows, we first recall the MCS approach. Then, using a motivating scenario, we propose some variations, enhancements, and extensions to basic MCSs. Finally, we shortly discuss the complexity of the enhanced framework and conclude. In the Appendix, we propose an example of use of the enhanced approach, in reference to the motivating scenario.

2 Bridge Rules and Multi-Context Systems

Heterogeneous MCSs have been introduced in [13] in order to integrate different inference systems without resorting to non-classical logical systems. Later, the idea has been further developed and generalized to non-monotonic reasoning domains —see, for instance, [3–6, 16] 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 heterogeneity of sources, each context is supposed to be based on its own *logic*, defined in a very general way [4]. 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. 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 de-

rived as consequences from a given $kb \in KB$. The set $\mathcal{C}n$ 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 $ACC \subseteq KB \times \mathcal{C}n$. Thus, $ACC(kb, S)$ means that belief set S is an acceptable consequence of knowledge base kb. We can also use ACC as a function $ACC : KB \to 2^{\mathcal{C}n}$ 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 $\langle KB, \mathcal{C}n, ACC \rangle$.

A *multi-context system* (MCS) $M = \{C_1, \ldots, C_n\}$ is a set of $n = |M|$ *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, or number i if omitted), $L_i = \langle KB_i, \mathcal{C}n_i, ACC_i \rangle$ is a logic, $kb_i \in KB_i$ a knowledge base, and br_i is a set of *bridge rules*, each of them $\rho \in br_i$ like

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

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, $1 \leq j \leq |M|$, and can be seen as a *query* to the (possibly different) context $C_j \in M$ 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 $(c_i : p)$. A *relational* MCS [12] is a variant where all the involved logics are relational, and aggregate operators in database style (like *sum*, *count*, *max*, *min*), are admitted in bridge-rule bodies.

A *belief state* (or *data state*) $\boldsymbol{S}$ of an MCS M is a tuple $\boldsymbol{S} = (S_1, \ldots, S_n)$ such that for $1 \leq i \leq n = |M|$, $S_i \in \mathcal{C}n_i$. 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 $\boldsymbol{S}$ when, for any $(c_j : p) \in body^+(\rho)$, $p \in S_j$, and for any $(c_k : p) \in body^-(\rho)$, $p \notin S_k$; so, $app(S)$ is the set of heads from those bridge rules *applicable* in S.

In managed MCSs (mMCSs)[4] 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 $F = \{s \in kb \,|\, kb \in KB\}$, 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 $F^{OP} = \{o(s) \,|\, o \in OP, s \in F\}$. 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[5]. 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

[4] For the sake of simplicity, we define mMCS simply over logics instead of "logic suite" as done in [5], where one can select the desired semantics among a set of possibilities.

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

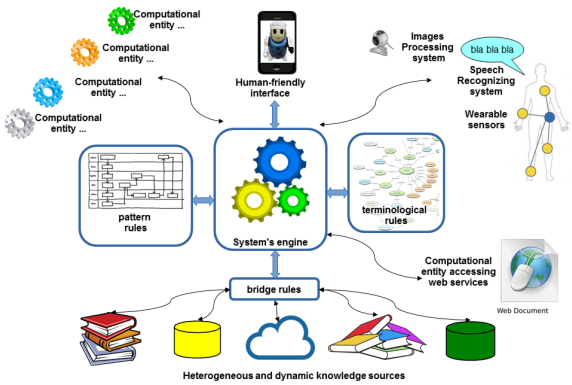

Fig. 1. F&K System: General Architecture

own management function mng_i which is crucial for knowledge incorporation from external sources and can be non-monotonic. 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 $\boldsymbol{S}$ is an *equilibrium* for an mMCS M iff, for $1 \leq i \leq |M|$,

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

Namely, we (i) apply all the bridge rules of C_i that are applicable in the belief state $\boldsymbol{S}$; (ii) apply the management function which, by incorporating bridge-rule results into C_i's knowledge base kb_i, computes a new knowledge base kb_i'; and, finally, (iii) determine 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 [3]; 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 [4, 10] but, in practice, they are only applicable when open-access to contexts' contents is granted. For practical purposes, one will 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 Proposed Extensions

Below we introduce some enhancements to the mMCS paradigm. We devised the proposed extensions in the perspective of the application of mMCSs to Cyber-Physical Systems (CPS). In Fig. 1 we depict the scenario outlined in [1], concerning "FRIENDLY & KIND with your Health" (F&K), kind Cyber-Physical Systems tailored for applications in the E-Health field. In an F&K system, an mMCS is considered to be an essential part of the system engine. Thus, for actual implementation of F&Ks and more generally of

intelligent logically-based CPS operating in the "Internet of Things", issues related to the concrete modalities of bridge-rule instantiation and execution need to be considered.

3.1 Update Actions 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* [6] and *evolving* [15] 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, [6] 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 [6, 15, 14]; however, it cannot be expressed *directly*; in view of practical applications, we introduce explicit and natural devices to cope with this important issue.

We proceed next to define a new extension called *timed* MCS (tMCS). In a tMCS, 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 [6]. 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*:

$$\mathcal{U}_i : KB_i \times 2^{act_i} \rightarrow 2^{KB_i} \setminus \{\emptyset\},$$

so that $kb'_i \in \mathcal{U}_i(kb_i, \Pi_i)$ means that the "new" knowledge base kb'_i is a possible result of updating kb_i with action Π_i under function $\mathcal{U}_i$. We thus assume $\mathcal{U}_i$ 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 $\mathcal{U}_i(kb_i, \Pi_i)$ is (possibly)

non-derterministic, but never empty (some resulting kb'_i is always specified). In some cases, we may even leave the knowledge base unaltered $kb_i \in \mathcal{U}_i(kb_i, \Pi_i)$. Notice 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 $\boldsymbol{\Pi}[T] = (\Pi_1[T], \ldots, \Pi_n[T])$. Finally, $S_i[T]$ denotes the set of beliefs for context C_i at instant T whereas, accordingly, $\boldsymbol{S}[T]$ denotes the belief state $(S_1[T], \ldots, S_n[T])$.

Definition 1 (timed context). *Let $C_i = \langle c_i, L_i, kb_i, br_i, OP_i, mng_i\rangle$ be an mMCS context. The corresponding* timed *context w.r.t. an initial belief state $\boldsymbol{S}$ is defined as:*

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

where $kb_i[0] := kb_i$ and $\boldsymbol{S}[0] := \boldsymbol{S}$, whereas $kb_i[T+1] := mng_i(app(\boldsymbol{S}[T]), kb')$ and $kb' \in \mathcal{U}_i(kb_i[T], \Pi_i[T])$.

Definition 2. *We let a tMCS at time T be $M[T] = \{C_1[T], \ldots, C_n[T]\}$.*

The initial timed belief state $\boldsymbol{S}[0]$ can possibly 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:

Definition 3 (timed equilibrium). *A timed belief state of tMCS M at time $T+1$ is a* timed equilibrium *iff, for $1 \leq i \leq n$ it holds that $S_i[T+1] \in ACC_i(kb_i[T+1])$.*

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 Def. 1, applicability is checked on belief state $\boldsymbol{S}[T]$, 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. [6] 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 [6] such an evolution is implicit in the notion of system run.

3.2 Bridge Rule Grounding and Activation

In the original definition of mMCSs, bridge rules are ground by definition. In [5], 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)". However, the practical method for obtaining such ground instances remains to be seen, especially since the basic definition of mMCS does not require either knowledge bases or bridge rules to be finite sets. Moreover, in the original definition, bridge-rule application is reactive: a bridge rule is applied whenever applicable. However, according to a context's own logic, other patterns of application might be defined, in principle; for instance, those introduced in [8].

In this section, we formally specify reasonable modalities for bridge-rules grounding and for proactive, rather than reactive, application. The issue of bridge-rule grounding has been discussed in [2] for relational MCSs where, however, the grounding of bridge rules is performed over a carefully defined finite domain, composed of constants only. As the authors observe, "*...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 restrictions on bridge-rule form. Such kind of limitation, which is common in logic programming approaches, prescribes that: (i) in the body of a rule, positive literals occur (in a left-to-right order) before negative literals, and their order is relevant; (ii) every variable occurring in the head of a non-ground bridge rule r also occurs in some positive literal of its body. We embrace the suggestion that bridge-rule grounding may occur at run-time, so we adopt the syntactic limitation. However, in bridge-rule grounding we intend to consider any, even infinite, domain, and we intend to consider timed belief states. In principle, bridge rules might be grounded over all terms that can be built over the signature of every context's underlying logic. This because mMCSs admit "value invention", i.e., via the results of bridge-rules application, beliefs (and their arguments) are propagated among contexts; so, via a bridge rule a context may obtain a result involving constants and terms previously not occurring therein. However, in each particular belief state it suffices to consider terms that actually occur in the composing belief sets: such sets are indeed those which determine bridge-rule applicability at that stage.

Definition 4. *Let $r \in br_i$ be a non-ground bridge rule in a context C_i of a given mMCS M with (timed) belief state $\boldsymbol{S}[T]$. A ground instance ρ of r w.r.t. $\boldsymbol{S}[T]$ is obtained by substituting every variable occurring in r with ground terms occurring*[6] *in $\boldsymbol{S}[T]$.*

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[7]

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

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

[6] Variables may occur in p of atoms $(c_j{:}p)$ in the body of r, in its head $o(s)$, or in both.

[7] Here, the dependency on T is fictitious. It will be relevant for bridge-rules patterns in Sec. 3.3.

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. $\boldsymbol{S}[T]$ of some non-ground r such that $hd(\rho) \in tr_i[T](kb_i[T])$.

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

Definition 6. *The set $app(\boldsymbol{S}[T])$ relative to ground bridge rules which are applicable in a timed data state $\boldsymbol{S}[T]$ of a given tMCS $M[T] = \{C_1[T], \ldots, C_n[T]\}$ is defined as:*

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

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 $T \geq T'$. 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.

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. Such a grounding, and thus the bridge-rule result, can be obtained, for example, by "executing" or "invoking" literals in the body (i.e., querying contexts) so as to obtain the grounding of positive literals first, to be extended 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*. Assuming that literals occurring in a bridge-rule body are ordered left to right as they appear in the rule definition we can talk about first, second, etc., positive/negative literal.

Definition 7. *A ground bridge rule $\rho \in grtr_i[T']$ for some T', of the form (1) 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 $\boldsymbol{S}[T] \models body(\rho')$.*

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:

Proposition 1. *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 $T_{k'}$.*

Proposition 2. *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, $hd(\rho) \in app(\boldsymbol{S}[T])$ iff for every negative literal $not\,(c_k{:}p)$ in the body, $h+1 \leq k \leq m$, $p \notin S_k[T]$.*

As the contexts composing an mMCS 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 state.

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 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 $\rho \in grtr_i[T']$ like (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 an mMCS 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 mMCS. 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,* $hd(\rho) \in app(\boldsymbol{S}[T_s])$ *and there exists* $T \leq T_m$ *such that* $hd(\rho) \in app(\boldsymbol{S}[T])$. *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. Thanks to the persistence of the system, fixing $T'_i = T_i$, for each $i \leq j$, suffices to prove the first property, and fixing $T = T_m$ 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 $r \in grtr_i[\hat{T}]$ 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.

3.3 Bridge-Rules Patterns

We now generalize and formalize for mMCS the enhanced form of bridge rules proposed in [9] for logical agents. In particular, we replace context names in bridge-rule bodies with *context designators*, which are special terms intended to denote a specific *kind* of context. For instance, a literal such as $doctorSmith{:}prescription(desease, P)$ asking family doctor, specifically Dr. Smith, for a prescription related to a certain desease, might become $family\_doctor(d){:}prescription(desease, P)$ 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 by 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.

Definition 8. *Let* C_i *be a context of an mMCS. A* context designator $m(k)$ *is a term where* m *is a fresh function symbol and* k *a fresh constant.*[8]

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

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

where each $\mathcal{C}_d$ *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 9. *Given an mMCS, each of the timed composing contexts* C_i, $1 \leq i \leq n$, *is defined, at time* 0, *as* $C_i[0] = \langle c_i, L_i, kb_i[0], br_i, brp_i, OP_i, mng_i, act_i, \mathcal{U}_i \rangle$, *where* brp_i *is a set of bridge-rule patterns, and all the other elements are as in Def. 1.*

Definition 10 (rule instance). *An* instance *of the bridge-rule pattern* $\phi \in brp_i$, *for* $1 \leq i \leq n$, *occurring in an mMCS, is a bridge rule* r *obtained by substituting every context designator occurring in* ϕ *by a context name* $c \in \{c_1, \ldots, c_n\}$.

[8] Meaning that m and k belong to the signature of L_i, but they do not occur in kb_i or in br_i.

The context names to replace a context designator must be established by suitable reasoning in context's knowledge base.

Definition 11. *Given an mMCS M, any composing context C_i, for $1 \leq i \leq n$, and a timed data state $\boldsymbol{S}[T]$ 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 in ϕ with a context name $c \in \{c_1, \ldots, c_n\}$ such that $S_i[T] \models subs_i(m_d(k_d), c)$, where $subs_i$ is a distinguished predicate that we assume to occur in the signature of L_i.*

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 12. *Given an mMCS, each of the timed composing contexts C_i, $1 \leq i \leq n$, is defined, at time $T+1$, as:*

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

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

All the other previously-introduced notions (equilibria, bridge-rule triggering and applicability, etc.) remain unchanged. We do not need to modify bridge-rule triggering, having already introduced the time parameter in the definition of the set $H_i[T]$ of bridge-rule heads at time T for context C_i. Notice 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 $subs_i(m_d(k_d), c)$ that can be defined so as to take several factors into account, among which, for instance, trust and preferences. Also, this enhancement goes in the direction of *dynamic* mMCS, where contexts can either join or leave the system during its operation. Such an extension has been in fact advocated since [4].

Definition 13. *We let a dynamic mMCS at time T be $M[T] = \{C_1[T], \ldots, C_{n_T}[T]\}$, where a* timed *belief state at time T is (as before) a tuple $\boldsymbol{S}[T] = (S_1[T], \ldots, S_{n_T}[T])$.*

The meaning is that the set of contexts which compose an mMCS may be different at different times. Here one can see the utility of having a constant acting as the context name; in fact, bridge-rule definition does not strictly depend upon the composition of M, as it would be by using integer numbers. Rather, applicability of a bridge rule depends on the presence in the system of contexts with the names indicated in the bridge-rule definition. Contexts can not only be added or removed, but also substituted by new "versions" with the same name.

3.4 Complexity

For space reasons we just 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 14. *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 $\boldsymbol{S}[t']$ at time $t' \leq t$.*

We resort, like [6], to *context complexity* as introduced in [11]. One has first to consider a *projected belief state* $\hat{\boldsymbol{S}}[t]$, 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 [2] for an algorithm which computes such sets of beliefs in the case of 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. The system context complexity of M is the smallest upper bound of the complexities of its contexts. The system context complexity depends upon the logics of the contexts in M.

The problems $Q^{\exists}$ and $Q^{\forall}$ are undecidable for infinite update sequences, because contexts' logics can in general simulate a Turing Machine and thus such problems reduce to the *halting* problem. Better complexity results can, however, be obtained under some restrictions. In particular, if we assume that all contexts C_i's knowledge bases and belief states are finite at any stage, all update functions $\mathcal{U}_i$, 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 proceed as follows: 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 $\hat{\boldsymbol{S}}[T]$ in some (all) timed equilibria $\boldsymbol{S}[T]$ 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 $\hat{\boldsymbol{S}}[T]$, 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. Therefore, for finite update sequences the context complexity determines the complexity of $Q^{\exists}$ and, complementarily, the complexity of $Q^{\forall}$.

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.

4 Concluding Remarks

In this paper we have proposed extensions to MCSs, aimed at practical application of such systems. The enhanced customizable bridge rules that we have formalized here have been succesfully experimented (though in a preliminar embryonic version) in a significant case-study, reported in [7]. We intend to employ the proposed features in the implementation, that will start in the near future, of an F&K Cyber-Physical System. This project will allow us to perform practical experiments in order to assess the

performance of this kind of systems, and to identify possible limitations and/or further aspects that can be subjected to improvements.

References

1. F. Aielli et al. FRIENDLY & KIND with your health: Human-friendly knowledge-INtensive dynamic systems for the e-health domain. In K. Hallenborg and S. Giroux, editors, *Proc. of A-HEALTH at PAAMS 2016.*, Comm. in Computer and Inf. Sci. Springer, 2016.
2. R. Barilaro, M. Fink, F. Ricca, and G. Terracina. Towards query answering in relational multi-context systems. In P. Cabalar and T. C. Son, editors, *Proc. of LPNMR 2013*, volume 8148 of *LNCS*, pages 168–173. Springer, 2013.
3. G. Brewka and T. Eiter. Equilibria in heterogeneous nonmonotonic multi-context systems. In *Proc. of the 22nd AAAI Conf. on Artificial Intelligence*, pages 385–390. AAAI Press, 2007.
4. G. Brewka, T. Eiter, and M. Fink. Nonmonotonic multi-context systems. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, volume 6565 of *LNCS*, pages 233–258. Springer, 2011.
5. G. Brewka, T. Eiter, M. Fink, and A. Weinzierl. Managed multi-context systems. In *Proc. of IJCAI 2011*, pages 786–791. AAAI, 2011.
6. G. Brewka, S. Ellmauthaler, and J. Pührer. Multi-context systems for reactive reasoning in dynamic environments. In T. Schaub, editor, *Proc. of ECAI 2014*. AAAI, 2014.
7. S. Costantini. ACE: a flexible environment for complex event processing in logical agents. In *EMAS 2015, Revised Selected Papers*, volume 9318 of *LNCS*. Springer, 2015.
8. S. Costantini. Knowledge acquisition via non-monotonic reasoning in distributed heterogeneous environments. In M. Truszczyński, G. Ianni, and F. Calimeri, editors, *Proc. of LPNMR 2013*, volume 9345 of *LNCS*. Springer, 2015.
9. S. Costantini and A. Formisano. Augmenting agent computational environments with quantitative reasoning modules and customizable bridge rules. In N. Osman and C. Sierra, editors, *AAMAS 2016 Best Workshop Visionary Papers*, volume 10003 of *LNCS*, 2016.
10. M. Dao-Tran, T. Eiter, M. Fink, and T. Krennwallner. Distributed evaluation of nonmonotonic multi-context systems. *J. of Artificial Intelligence Research*, 52:543–600, 2015.
11. T. Eiter, M. Fink, P. Schüller, and A. Weinzierl. Finding explanations of inconsistency in multi-context systems. In F. Lin, U. Sattler, and M. Truszczyński, editors, *Proc. of KR 2010*. AAAI, 2010.
12. M. Fink, L. Ghionna, and A. Weinzierl. Relational information exchange and aggregation in multi-context systems. In *Proc. of LPNMR 2011*, volume 6645 of *LNCS*, pages 120–133, 2011.
13. F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics or: How we can do without modal logics. *Artif. Intell.*, 65(1):29–70, 1994.
14. R. Gonçalves, M. Knorr, and J. Leite. Evolving bridge rules in evolving multi-context systems. In *Proc. of CLIMA 2014*, volume 8624 of *LNCS*, pages 52–69. Springer, 2014.
15. R. Gonçalves, M. Knorr, and J. Leite. Evolving multi-context systems. In *Proc. of ECAI 2014*, pages 375–380. IOS, 2014.
16. F. Roelofsen, L. Serafini, and A. Cimatti. Many hands make light work: Localized satisfiability for multi-context systems. In R. López de Mántaras and L. Saitta, editors, *Proc. of ECAI 2004*, pages 58–62. IOS Press, 2004.

A Appendix: An Example

In this Section we provide an example of use to illustrates the new features. We refer to Example 2 in [6] where it is supposed that a user, Bob, suffering from dementia, is able to live at home as an assisted living system, modeled by means of an rMCS, is able to constantly track Bob's position, and to detect and take care of emergencies (such as Bob's forgetting the stove on), 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 but also to notice anomalous behavior which might signify an impaired state. The input can be provided by means of sensors, but also by recording and monitoring Bob's activities via suitable telemedicine appliances, via, e.g., videocameras or via a personalized assistant/monitoring agent (PMA) which might be incarnated, in different use-cases, in a chatterbot/avatar on tv or on the mobile phone, or even in a social robot; in our view, the 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. The patient's PMA (in this case Bob's), is assumed below to be modeled as a context included in an mMCS where such context is in particular an agent defined in any agent-oriented logic language [8].

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 an hemorrhage, immediate help is asked for. Clearly, the example is over-simplified and we do not aim at medical accuracy.

Following [6], 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 mentioned before, literals in bridge-rule bodies with no indication of a context are proved locally.

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.

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

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 prompltly 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 cardiophatic 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.

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

In case a patient's health state is not really critical but is anyway altered, a physician must be consulted. However, in case, e.g., 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. 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.

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

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.

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

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:

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