Bridge Rules for Reasoning in Component-Based Heterogeneous Environments

Stefania Costantini and Giovanni De Gasperis

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica,
Università degli Studi dell'Aquila, Via Vetoio snc, L'Aquila, ITALY

Abstract. Multi-Context Systems (MCS) 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 of such systems; in particular, we discuss aspects that might be improved to increase practical applicability.

1 Introduction

Multi-Context Systems (MCSs) have been proposed in Artificial Intelligence and Knowledge Representation to model information exchange among several diverse sources [1, 2, 3]. MCSs are designed so as to deal with heterogeneous sources: in fact, the approach explicitly considers their different representation languages and semantics. Heterogeneous sources are called "contexts" (or, equivalently, we will call them "sources", or "modules"), and interact through special inter-context rules called *bridge rules*, similar in format to datalog rules with negation [1].

The reason why MCSs are particularly interesting is that they aim at modeling in a formal way real applications requiring access to sources distributed on the web. Among the relevant domains where the adoption of MCSs can bring real advances is for instance health care (see, e.g., the running example in [6]). In view of such practical applications it is important to notice that, being logic-based, contexts may encompass logical agents, to which MCSs have in fact already been extended (cf. [7, 8]).

Despite the importance of MCSs for practical knowledge representation and reasoning, their definition is under some aspects too abstract, and the functioning modalities of such systems are considered under ideal circumstances. In this paper we try to tackle in a formal way the practical aspects related to these systems, and attempt at a systematization that should also provide guidelines for implementations. The paper proposes some substantial technical improvements concerning bridge rules, also in relation to the evolution of an MCS over time.

The paper is organized as follows. In Section 2 we introduce Multi-Context Systems. In Section 3 we propose a motivating application scenario, and with respect to such scenario we outline some aspects where the original MCS definition is not fully adequate in practice. In Section 4 we propose some variations, enhancements and extensions to the basic approach, that introduce improvements concerning these aspects. Finally, in Section 5 we conclude.

[1] cf. [4, 5] for standard datalog, logic programming and prolog terminology.

2 Bridge Rules and Multi-Context Systems: Background

Heterogeneous Multi-Context systems have been introduced in the seminal work of [9] 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 in [1, 2, 6, 3] and other related papers. There, (managed) Multi-Context systems aim at making it possible to build systems that need to access multiple possibly heterogeneous data sources, called "contexts", by modeling the necessary information flow via "bridge rules", whose form is similar to datalog rules with negation (cf., e.g., [5]). Bridge rules allow for inter-context interaction: in fact, each element in their "body" explicitly includes the indication of the context from which information is to be obtained.

In order to account for heterogeneity, each context is supposed to be based on its own *logic*. Reporting from [2], a logic L is a triple $(KB_L; Cn_L; ACC_L)$, where KB_L is the set of admissible knowledge bases of L, that are sets of KB-elements ("formulas"); underlying (though here implicitly) there is a signature Σ_L including sets of constants, predicate and function symbols, and a set of variables; KB_L elements are thus specified over this signature and involve terms that can be either variables or constants or compound terms built out of function symbols and other terms; atoms are defined as the application of a predicate over a set of terms, according to the predicate's arity; 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. Cn_L is the set of possible sets of consequences of knowledge bases in KB_L; sets in Cn_L can be called "belief sets" or "data sets", as their elements are data items or "beliefs" or "facts", that we assume to be ground. The function $ACC_L : KB_L \rightarrow 2^{Cn_L}$ defines the semantics of L by assigning to each knowledge base "acceptable" sets of consequences; so, only some (or possibly none) of the possible sets of consequences in Cn_L are acceptable.

A multi-context system (MCS) $M = (C_1, \ldots, C_n)$ is a collection of contexts $C_i = (L_i; kb_i; br_i)$ where L_i is a logic, $kb_i \in KB_{L_i}$ is a knowledge base and br_i is a set of bridge rules. Each such rule ρ is of the following form, where the left-hand side s is called the *head*, denoted as $hd(\rho)$, the right-hand side is called the *body*, also denoted as $body(\rho)$, and the comma stand for conjunction.

$s \leftarrow (c_1 : p_1), \ldots, (c_j : p_j), not\,(c_{j+1} : p_{j+1}), \ldots, not\,(c_m : p_m).$

For each bridge rule included in context C_i the head s can be any formula in L_i. It is required that $kb_i \cup s$ belongs to KB_{L_i} and, for every $k \leq m$, c_k is a constant denoting a context included in M (in the original definitions c_k is simply be an integer number $i \leq n$, though more expressive "names" can be used), and each p_k belongs to some set in Cn_{L_k}, i.e., it is a possible consequence of context c_k's knowledge base according to the logic in which c_k is defined. The head s is any formula in L_i, where however $kb_i \cup \{s\} \in KB_{L_i}$. A *relational* MCS [10] is a variant where all the involved logics are relational, and aggregate operators in database style are admitted in bridge-rule bodies.

A data state of MCS M is a tuple $S = (S_1, \ldots, S_n)$ such that for $1 \leq i \leq n$, $S_i \in Cn_{L_i}$. Thus, a data state associates to each context a possible set of consequences.

Given data state S, $app(S)$ is the set composed of the heads of those bridge rules which are *applicable* in S as their body is entailed by S; i.e., those such that for every positive literal $(c_i : p_i)$ in the body, $1 \leq i \leq j$, $p_i \in S_i$ and for every negative literal $not\,(c_k : p_k)$ in the body, $j+1 \leq k \leq m$, $p_k \notin S_k$.

In managed MCSs (mMCSs)[2] the conclusion s, which represents the "bare" bridge-rule result, becomes $o(s)$ where o is a special operator. The meaning is that the result computed by a bridge rule is not blindly incorporated into the "destination" context's knowledge base: rather, it is processed by operator o, that can possibly perform any elaboration, such as format conversion, belief revision, etc.

More precisely, for given logic L, $F_L = \{s \in kb \,|\, kb \in KB_L\}$ is the set of formulas occurring in its knowledge bases. A *management base* is a set of operation names (briefly, operations) OP, defining elaborations that can be performed on formulas, e.g., addition of, revision with, etc. For a logic L and a management base OP, the set of operational statements that can be built from OP and F_L is $F_L^{OP} = \{o(s) \,|\, o \in OP, s \in F_L\}$. The semantics of such statements is given by a *management function*, which maps a set of operational statements and a knowledge base into a modified knowledge base. In particular, a management function over a logic L and a management base OP is a function $mng : 2^{F_L^{OP}} \times KB^L \rightarrow 2^{KB_L} \setminus \emptyset$. We assume a management function to be deterministic, i.e., to produce a unique new knowledge base. Each context in an mMCS has its specific management function mng_i, which is crucial for knowledge incorporation from external sources. Notice that each mng_i can be non-monotonic, i.e., it may imply deletion of formulas. Now, we can see a context as $C_i = (c_i; L_i; kb_i; br_i; OP_i; mng_i)$ where c_i is a constant acting as the context "name" that, if omitted, is assumed to be integer number i.

Desirable data states, called *equilibria*, are those which encompass bridge-rules application. In fact in (m)MCSs equilibria are those data states S where each S_i is acceptable according to function ACC_i associated to L_i, given that every applicable bridge rule has indeed been applied. Formally, a data state S is an equilibrium for an MCS iff, for $1 \leq i \leq n$,

$$S_i \in ACC_i(mng_i(app(S), kb_i)) \quad (1)$$

I.e., one (i) applies all C_i's bridge rules which are applicable in data state 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'; (iii) determines via ACC_i the set of acceptable sets of consequences of kb_i'. In an equilibrium such set includes S_i, i.e., an equilibrium is "stable" w.r.t. bridge-rule application.

Conditions for existence of equilibria have been studied [1], and basically require cyclic application of bridge rules to be avoided. The complexity of deciding whether some equilibrium exists depends upon composing contexts' complexity, basically upon the complexity of computing formula (1).

Algorithms for computing equilibria have recently been proposed [2, 11, 12]. Methods also exist [6] to detect and enforce MCS's consistency, i.e., to ensure that an equilibrium does not include inconsistent data sets (*local consistency*) and that the composing

[2] We introduce mMCSs in a simplified form with respect to [6]: in fact, they generalize from a logic to a "logic suite", where one can select the desired semantics among a set of possibilities, while we define mMCS simply over logics.

data sets are mutually consistent (*global consistency*). It has been proved that *local consistency* is achieved whenever all management functions are (lc-) preserving, i.e., if they always determine a kb' which is consistent.

Bridge rules as defined in mMCSs are basically a *reactive* device, as a bridge rule is applied whenever applicable. In dynamic environments, 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, contexts may be able to incorporate new data items, e.g, as discussed in [3] for Reactive MCSs (rMCSs), the input provided by sensors ("observations"). Therefore, a bridge rule can be in principle re-evaluated upon new observations, thus leading to evolving equilibria and to the notion of a "run" of an rMCS.

3 Motivating Scenario 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 [13], 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 is depicted in Figure 1.

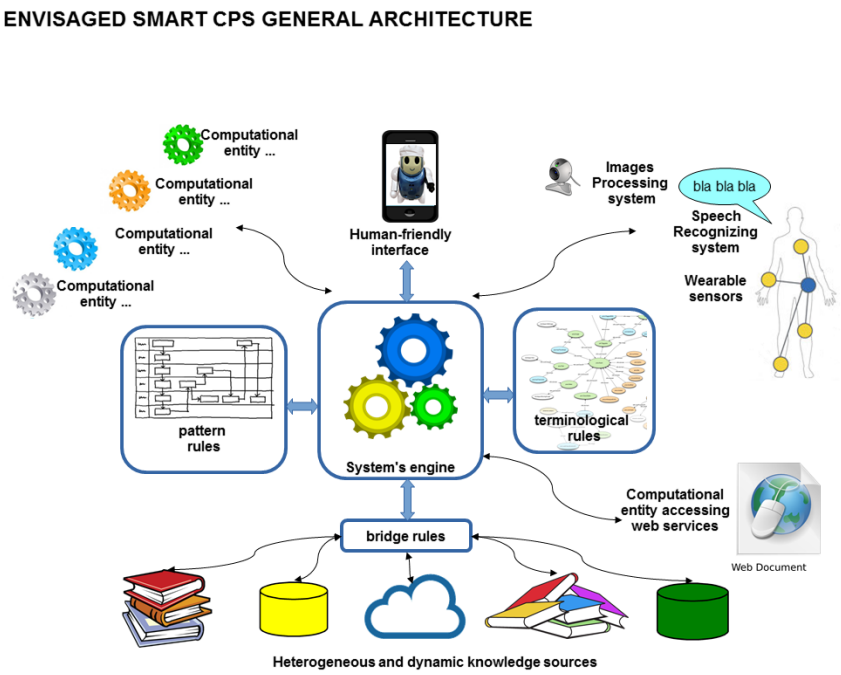

Fig. 1. Motivating Scenario

We have a set of computational entities, of knowledge bases and of sensors, all immersed in the "Fog" of the Internet of Everything. 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. The System's engine will keep track of the 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. 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.

In the perspective of such kind of systems, the definition of (m)MCS recalled in Section 2 is, though neat, quite abstract. Some limitations can be identified, that we list below.

Grounded Knowledge Assumption. Bridge rules are by definition ground, i.e., they do not contain variables. In [6] 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)*". Basic definition of mMCS do 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 [14]. 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. The issue of bridge-rule grounding has been discussed in [15] for relational MCSs, where however the grounding is performed over a carefully defined finite domain, composed of constants only.

Logical Omniscience and Unbounded Resources Assumption. A bridge rule is supposed to be applied whenever its body is entailed by the current data state. However, contexts will hardly compute their full set of consequences beforehand. So, practical bridge rule application will presumably consist in posing queries to other contexts which are situated somewhere in the nodes of a distributed systems. Each source will need time to compute and deliver the required result, and might even never be able do so, in case of reasoning with limited resources or of network failures.

Update Problem. Considering inputs from sensor networks as done in [3] is a starting point: however, sources can be updated in many ways via the interaction with their environment. For instance, agents are supposed to continuously modify themselves via the

interaction with the environment, but even a plain relational database can be modified by its users/administrators.
Static system Assumption. The definition of mMCS might realistically be extended to a setting where the set of contexts changes over time, maybe because some context gets momentarily disconnected, or because components may freely either join or abandon the system. Moreover inter-context reachability might be limited, e.g., via authorizations of some kind.
Full System Knowledge Assumption. A context might know the *role* of another context it wants to query (e.g., a 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.
Unique Source Assumption. In the body of bridge rules, each literal mentions a specific context. In practice, that context might not be able to return a result while another context with the same role instead might.
Uniform Knowledge Representation Format Assumption. Different contexts might represent similar concepts in different ways: this aspect is taken into account in [8], where ontological definitions can be exchanged among contexts, and a possible global ontology is also considered.
Equilibria Computation and Consistency Check Assumption. 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.

4 Proposed Extensions

Below we consider the points raised in previous section and provide, whenever not already existing, related extensions/enhancements to the basic mMCS paradigm.

4.1 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. The issue has been discussed in [15] for relational MCSs, where however the grounding of bridge rules is performed over a carefully defined finite domain, composed of constants only. Instead, we intend to consider any, even infinite, domain.

The procedure for computing equilibria that we propose for the case of non-ground bridge rules is, informally, the following. (i) We consider an initial data state S_0 composed of finite sets; this is without loss of generality because, as seen below, it does not actually limit the grounding to finite domains. (ii) We instantiate bridge rules over the finite number of (ground) terms occurring in S_0; we thus obtain an initial finite 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 start the procedure from a basic data state consisting of finite ground instances of the initial contexts' knowledge bases, obtained by substituting variables with constants. By definition, a ground instance of a context's C_i knowledge base is in fact in Cn_i, i.e., it is indeed a set of possible consequences, though in general it is not acceptable. Notice that starting from a finite data state does not guarantee however neither the existence of a finite equilibrium, nor that an equilibrium can be reached in a finite number of steps.

Consider as an example an MCS composed of two contexts C_1 and C_2, both based upon plain logic programming and 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).

$\%kb_1$

$nat(0).$

$\%kb_2$

$nat(suc(X)) \leftarrow nat(X).$

$\%br_2$

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

The unique equilibrium is reached in one step from basic data state $S_0 = (\{nat(0)\}, \emptyset)$ via the application of br_2 which "communicates" fact $nat(0)$ to C_2. In fact, due to the the recursive rule, we have the equilibrium (S_1, S_2) where
$S_1 = \{nat(0)\}$ and $S_2 = \{nat(0), nat(suc(0)), nat(suc(suc(0))), \ldots\}\rangle$
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.

The next example is a variation of the former one where C_1 "produces" the even natural numbers (starting from 0) and C_2 the odd ones. There is clearly a unique equilibrium, that cannot however be reached in finite time.

$\%kb_1$

$nat(0).$

$\%br_1$

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

$\%kb_2$

$\emptyset$

$\%br_2$

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

We may notice that the contexts in the above example enlarge their knowledge by means of mutual "cooperation". Let us consider, according to our proposed method, again the basic data state $S_0 = (\{nat(0)\}, \emptyset)$.
As stated above, we ground bridge rules on the terms occurring therein. S_0 is not an equilibrium for the given MCS: in fact, the bridge rule in kb_2, once grounded on constant 0, is applicable but not applied. The data set resulting from the application, i.e.,

$S' = (\{nat(0)\}, \{nat(suc(o))\})$ is not an equilibrium either, because now the bridge rule in kb_1 (grounded on $suc(0)$) is in turn applicable but not applied.

We may go on, as $S'' = (\{nat(0), nat(suc(suc(0)))\}, \{nat(suc(o))\})$ leaves the bridge rule in kb_2 to be applied (grounded on $suc(suc(0))$), and so on. The unique equilibrium, that cannot be reached in finite time, is composed of two infinite sets, the former one representing the even natural numbers (including zero) and the latter representing the odd natural number. The equilibrium may be represented as:
$E = (\{nat(0), nat(suc^k(0))),\ k \bmod 2\ = 0\}, \{nat(suc^k(o)),\ k \bmod 2\ = 1\})$

We have actually devised and applied an adaptation to non-ground bridge rules of the operational characterization introduced in [1] for the grounded equilibrium of a *definite* MCS, as 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 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. We require a minor modification in bridge-rule syntax: we assume in particular 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 assume that p_j is proved locally from the present context's knowledge base. Previous example can be reformulated as follows, where we assume the customary prolog's syntax, and prolog's procedural semantics where elements in the body of a rule are proved/executed left-to-right. The knowledge bases and bridge rules now are:

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

In the new definition there is a counter (initialized to zero) and some threshold, say t. We will 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 [16].

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

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

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 [1] 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 [1], for simplicity we assume bridge-rules bodies 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 S. A ground instance ρ of r w.r.t. 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 S.*

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

Definition 2. *The set $app(S)$ relative to ground bridge rules which are applicable in a data state S of a given mMCS $M = (C_1, \ldots, C_n)$ is now defined as follows.*
$app(S) = \{hd(\rho) \mid \rho$ *is a ground instance w.r.t. S of some bridge rule $r \in br_i, 1 \leq i \leq n$, and* $app^{\models}{}_g(\rho, S) = \mathsf{true}\}$

We assume, analogously to [1], that given mMCS is *monotonic*, which here means that for each C_i: (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 context C_i, function ACC'_i be a variation of ACC_i which selects one single set E_i among those generated by ACC_i. I.e., given context C_i and knowledge base $\hat{kb} \in KB_{L_i}$, $ACC'_i(\hat{kb}) = E_i$ where $E_i \in ACC_i(\hat{kb})$. Let ∞ be the first infinite ordinal number isomorphic to the natural numbers.

Definition 3. *Consider mMCS $M = (C_1, \ldots, C_n)$ with no negative literals in bridge-rule bodies, and assume arbitrary choice of function ACC'_i for each composing context C_i. Let, for $1 \leq i \leq n$, $gr(kb_i)$ be the grounding of kb_i w.r.t. the constants occurring in any kb_j, $1 \leq j \leq n$. A data state of grade κ is obtained as follows.*
For $i \leq n$ and $\alpha = 0$, we let $kb_i^0 = gr(kb_i)$, and we let $S^\alpha = S^0 = (kb_1^0, \ldots, kb_n^0)$
For each $\alpha > 0$, we let $S^\alpha = (S_1^\alpha, \ldots, S_n^\alpha)$ and $S_i^\alpha = ACC'_i(kb_i^\alpha)$
where for finite κ and $\alpha \geq 0$ we have
$kb_i^{\alpha+1} = mng_i(app(S^\alpha), kb_i^\alpha)$ *if* $\alpha < \kappa$,
$kb_i^{\alpha+1} = kb_i^\alpha$ *otherwise*
while if $\kappa = \infty$ we have $kb_i^\infty = \bigcup_{\alpha \geq 0} kb_i^\alpha$

Differently from [1], 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 [1] 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 $M = (C_1, \ldots, C_n)$ be a monotonic mMCS with no negative literals in bridge-rule bodies. A belief state $S = (S_1, \ldots, S_n)$ is a grounded equilibrium of grade κ of M iff $ACC'_i(mng_i(app(S), kb_i^\kappa) = S_i$, for $1 \leq i \leq n$.*

Several grounded equilibria may exist, depending upon the choice of ACC'_i. The required grade for obtaining an equilibrium would be $\kappa = \infty$ in the former version of the example, where in the latter version if setting threshold t we would have $\kappa = t$. We can state the following relationship with [1]:

Proposition 1. *Let $M = (C_1, \ldots, C_n)$ be a definite MCS (in the sense of [1]), and let $S = (S_1, \ldots, S_n)$ be a grounded equilibrium for M, reachable in δ steps. Then, there exists a choice of function ACC'_i for each context C_i of M such that S is a grounded equilibrium of grade δ for the mMCS M' obtained from M by choosing, for $i \leq n$, a management function mng_i that just adds to kb_i every s such that $o(s) \in app(S)$.*

In an implemented mMCS, as remarked in [15], "...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.

Also, we assume that bridge-rule application is not necessarily reactive but that, according to a context's own logic, other modalities of application may exist; for instance, the modalities introduced in [7, 8] cope with "Logical Omniscience and Unbounded Resources Assumption" by detaching (proactive) bridge-rule application from the processing of the management function. Thus, in our case 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 for instance by "executing" or "invoking" literals in the body (i.e., querying contexts) left-to-right in prolog style. In practice, we can 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, i.e., we assume that *(i) every variable occurring in the head of a non-ground bridge rule r also occurs in some positive literal of its body; and (ii) in the body of such 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. Each positive literal $(c_i : p_i)$ in the body

may fail (i.e., c_i will return a negative answer), if none of the instances of p_i given the partial instantiation computed so far is entailed by c_i's present data state. Otherwise, the literal succeeds and subsequent ones are instantiated to its results. Negative literals $not\,(c_j : p_j)$ make sense only if p_j is ground at the time of invocation, and succeed if p_j is *not* entailed by c_j's present data state. In case either some literal fails or a non-ground negative literal is encountered, 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. It is easy to prove that the invocation of a bridge rule leads to success if and only if, given its ground instance obtained via the above-specified evaluation pattern, the body is entailed by the present system's data state (which is hopefully an equilibrium) and thus the rule is applicable (according to the previously-reported notions of applicability). We omit formal definitions and proofs for lack of space. However, we may notice that asynchronous application of bridge rules determine evolving equilibria.

4.2 Update Problem

In dynamic environments, contexts are in general able to incorporate new data items, e.g, as discussed in [3], the input provided by sensors. We intend to explicitly take into account not only sensor input, but more generally the interaction of contexts with an external environment. As a premise we assume, similarly to what is done in Linear Time Logic (LTL), a discrete, linear model of time where each state/time instant can be represented by an integer number. States $t_0, t_1, \ldots$ can be seen as time instants (or 'time points') in abstract terms, though in practice we have $t_{i+1} - t_i = \delta$, where δ is the actual interval of time after which we assume a given system to have evolved.

We assume then that each context is subjected at each time point to a (possibly empty) finite update. Thus, for mMCS $M = (C_1, \ldots, C_n)$ let $\Pi_T = \langle \Pi_T^1, \ldots \Pi_T^n \rangle$ be a tuple composed of the finite updates performed to each module at time T, where for $1 \leq i \leq n$ Π_T^i is the update to C_i. Let $\Pi = \Pi_1, \Pi_2, \ldots$ be a sequence of such updates performed at time instants $t_1, t_2, \ldots$. Let us assume that each context copes with updates in its own particular way, so let $\mathcal{U}_i$, $1 \leq i \leq n$ be the *update operator* that module C_i employs for incorporating the new information, and let $\mathcal{U} = \{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$ be the tuple composed of all these operators. We assume $\mathcal{U}_i$ to encompass all possible updated performed to a module, included sensor input. So (analogously to the management function) let the *update base* $uops_i$ be a set of update operations which are admitted on context C_i. Then we have: $\mathcal{U}_i : 2^{uops} \times KB^L \rightarrow 2^{KB_L} \setminus \emptyset$. Notice that updates can be non-monotonic.

Consequently, we allow contexts' knowledge bases and data states to evolve in time: a *timed* data state at time T is a tuple $S^T = (S_1^T, \ldots, S_n^T)$ such that each S_i^T is an element of Cn_i at time T. We assume the timed data state S^0 to be an equilibrium according previous definitions. Later on however, transition from a timed data 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. An mMCS at time 0 is as defined previously, while at time $T + 1$ its knowledge base, and thus its data states and equilibria, will have evolved, where also the notion of bridge-rule applicability is now performed according to Definitions 1 and 2, but relatively to a timed data state S^T.

Therefore, by letting, for each C_i $i \leq n$, $kb_i^0 = kb_i$ we have that

Definition 5. *A timed data state of mMCS M at time $T+1$ is an equilibrium iff, for $1 \leq i \leq n$,*

$$S_i^{T+1} \in ACC_i(mng_i(app(S^T), kb_i^{T+1}))$$

where $kb_i^{T+1} = \mathcal{U}_i(kb_i^T, \Pi_T^i)$.

The meaning is that an equilibrium is now a data state which encompasses bridge rules applicability on the updated contexts' knowledge bases. Notice that, in practice, for each bridge rule applicable at time T the state when its result will actually affect the destination context is in general unpredictable. In fact, contexts occurring in bridge-rule bodies will require some amount of time for returning their results.

4.3 Static System and Full System Knowledge Assumption

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) that 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 context: rather, there might be specific authorizations involved. These aspects may be modeled by means the following extensions:

Definition 6. *A* dynamic *managed Multi-Context System (dmMCS) at time T is a set $M^T = (C_1, \ldots, C_n, Dir, Reach)$ of contexts where $M = (C_1, \ldots, C_n,)$ is an mMCS and Dir and $Reach$ are special contexts without associated bridge rules where:*

- *Dir is a* directory *which contains the list of the contexts, namely $C_1, \ldots, C_n$, 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.*

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

Definition 7. *Given a dmMCS (at time T) M^T, each (non-ground) bridge rule r in the composing contexts $C_1, \ldots, C_n$ has the form:*

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

where for $1 \leq k \leq m$ the expression $\mathcal{C}_k$ 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 expressions $role_k@Dir$ 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 8. *Let M^T be a dmMCS (at time T) and S^T be a timed data state for M^T. Let r be a bridge rule in the form specified in Definition 7. 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 9. *Let r' be a pre-ground version of a bridge rule r occurring in context $\hat{C}$ of dmMCS M^T (at time T) with timed data state S^T. Let ρ be a ground instance w.r.t. S^T of r'. We have now $hd(\rho) \in app(S^T)$ if ρ fulfills the conditions for applicability w.r.t. S^T 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.

4.4 Unique Source Assumption

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 dmMCS, have the specified role. So, the extension that we propose in this section can be called a *multi-source option*. In particular, for dmMCS M^T, composed at time T of contexts $C_1, \ldots, C_n$, the expression $role_k@Dir$ occurring in bridge rule $r \in br_s$ will now denote some nonempty set $SC_k \subseteq (\{C_1, \ldots, C_n\} \setminus \{C_s\})$, indicating the contexts with the required role (where C_s is excluded as a context would not look for 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 10. *Let M^T be a dmMCS (at time T) and S^T be a timed data state for M^T. Let $r \in br_s$ be a bridge rule in the form specified in Definition 7 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 9, 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.

Definition 11 (Preferred Source Selection). *Given a query $role@Dir$ with result SC, a preference criterion $\mathcal{P}$ returns a (nonempty) ordered subset $SC^{\mathcal{P}} \subseteq SC$.*

Different preference criteria can be defined according to several factors such as trust, reliability, fast answer, and others. Approaches to preferences in logic programming might be adapted to the present setting: cf., among many, [3] and the references therein, [17, 18] and [19, 19]). The definition of a context will now be as follows.

Definition 12. *A context C_i included in a dmMCS (except for Dir and $Reach$) is defined as $C_i = (L_i; kb_i; br_i; \mathcal{P}_i)$ where L_i, kb_i and br_i are as defined before, and $\mathcal{P}_i$ is a preference criterion as specified in Definition 11.*

5 Concluding Remarks

In this paper we have discussed and extended mMCSs, which are a general and powerful framework for modeling systems composed by several heterogeneous and possibly distributed sources (contexts), that interact via so-called bridge rules. The proposed extensions improve practical applicability of mMCSs by: making bridge rules more general and flexible; introducing explicit time so as to model contexts' updates and consequent system's evolution; introducing concepts of inter-context reachability and contexts' role, and preferences among reachable contexts with desired role. We believe that implementations of mMCSs might profit from the enhancements that we have introduced here.

Future work involves in fact the implementation as an mMCS of a smart Cyber-Physical System in the e-Health domain for intelligent monitoring of patients with comorbidities [13]. This will allow us to experiment, refine and further develop the new features.

References

[1] Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: Proc. of the 22nd AAAI Conf. on Artificial Intelligence, AAAI Press (2007) 385–390

[2] Brewka, G., Eiter, T., Fink, M.: Nonmonotonic multi-context systems: A flexible approach for integrating heterogeneous knowledge sources. In Balduccini, M., Son, T.C., eds.: Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning - Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday. Volume 6565 of Lecture Notes in Computer Science., Springer (2011) 233–258

[3] Brewka, G., Ellmauthaler, S., Pührer, J.: Multi-context systems for reactive reasoning in dynamic environments. In Schaub, T., ed.: ECAI 2014, Proc. of the 21st European Conf. on Artificial Intelligence, IJCAI/AAAI (2014)
[4] Lloyd, J.W.: Foundations of Logic Programming. Springer-Verlag (1987)
[5] Apt, K.R., Bol, R.N.: Logic programming and negation: A survey. The Journal of Logic Programming **19-20** (1994) 9–71
[6] Brewka, G., Eiter, T., Fink, M., Weinzierl, A.: Managed multi-context systems. In Walsh, T., ed.: IJCAI 2011, Proc. of the 22nd Intl. Joint Conf. on Artificial Intelligence, IJCAI/AAAI (2011) 786–791
[7] Costantini, S.: Knowledge acquisition via non-monotonic reasoning in distributed heterogeneous environments. In Truszczyński, M., Ianni, G., Calimeri, F., eds.: 13th Int. Conf. on Logic Programming and Nonmonotonic Reasoning LPNMR 2013. Proc. Volume 9345 of Lecture Notes in Computer Science., Springer (2015)
[8] Costantini, S., De Gasperis, G.: Exchanging data and ontological definitions in multi-agent-contexts systems. In Paschke, A., Fodor, P., Giurca, A., Kliegr, T., eds.: RuleMLChallenge track, Proceedings. CEUR Workshop Proceedings, CEUR-WS.org (2015)
[9] Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics or: How we can do without modal logics. Artif. Intell. **65**(1) (1994) 29–70
[10] Fink, M., Ghionna, L., Weinzierl, A.: Relational information exchange and aggregation in multi-context systems. In Delgrande, J.P., Faber, W., eds.: Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Proceedings. Volume 6645 of Lecture Notes in Computer Science., Springer (2011) 120–133
[11] Dao-Tran, M., Eiter, T., Fink, M., Krennwallner, T.: Distributed evaluation of nonmonotonic multi-context systems. JAIR, the Journal of Artificial Intelligence Research **52** (2015) 543–600
[12] Eiter, T., Simkus, M.: Linking open-world knowledge bases using nonmonotonic rules. In Truszczyński, M., Ianni, G., Calimeri, F., eds.: 13th Int. Conf. on Logic Programming and Nonmonotonic Reasoning LPNMR 2013. Proc. Volume 9345 of Lecture Notes in Computer Science., Springer (2015)
[13] 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 Hallenborg, K., Giroux, S., eds.: International Workshop on Agents and multi-agent Systems for AAL and e-HEALTH (A-HEALTH) at PAAMS 2016. Proceedings. Communications in Computer and Information Science, Springer (2016)
[14] Gelfond, M.: Answer sets. In: Handbook of Knowledge Representation. Elsevier (2007)
[15] Barilaro, R., Fink, M., Ricca, F., Terracina, G.: Towards query answering in relational multi-context systems. In Cabalar, P., Son, T.C., eds.: Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Proceedings. Volume 8148 of Lecture Notes in Computer Science., Springer (2013) 168–173
[16] 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, Springer-Verlag, Berlin (2002) 50–61
[17] 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). (2010) 414–424
[18] Brewka, G., Niemelä, I., Truszczyński, M.: Preferences and nonmonotonic reasoning. AI Magazine **29**(4) (2008)
[19] Costantini, S., Formisano, A.: Modeling preferences and conditional preferences on resource consumption and production in ASP. Journal of of Algorithms in Cognition, Informatics and Logic **64**(1) (2009)