Information Flow within Relational Multi-context Systems

Luís Cruz-Filipe,Graça Gaspar and Isabel Nunes

DI–FCUL–TR–2014–03

DOI:10455/6900

(http://hdl.handle.net/10455/6900)

October 2014

Information Flow within Relational Multi-context Systems

Luís Cruz-Filipe[1], Graça Gaspar[2], and Isabel Nunes[2]

[1] Dept. of Mathematics and Computer Science, University of Southern Denmark
[2] LabMAg, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract. Multi-context systems (MCSs) are an important framework for heterogeneous combinations of systems within the Semantic Web. In this paper, we propose generic constructions to achieve specific forms of interaction in a principled way, and systematize some useful techniques to work with ontologies within an MCS. All these mechanisms are presented in the form of general-purpose design patterns. Their study also suggests new ways in which this framework can be further extended.

1 Introduction

In parallel with the proliferation of different reasoning systems, larger and larger bodies of knowledge are being built in several fields, each with its expressiveness and efficiency, that can benefit enormously from adequate frameworks allowing to reason with information coming from different sources. Integrating several knowledge sources in a modular and flexible way is nowadays a growing need, and there has been significant growth in the research and development of this kind of heterogenous systems. As such, best practices should be devised as early as possible to guide the design and implementation of these systems, as has been done for other frameworks [2, 20, 33].

A particular class of such heterogeneous combinations is that of non-monotonic multi-context systems (MCSs) [3], which consist of several independent systems (the "contexts") interacting through Datalog-style "bridge rules", controlling how information flows by means of knowledge that is added to a context whenever some information can be inferred from other contexts. MCSs have been a topic of active research in recent years, with effort being put in addressing specific problems of this framework, and several variants of MCSs have been proposed to deal with particular situations. Of particular interest are *relational* multi-context systems [16], where each context is allowed to have a first-order sublanguage freely generated from a given set of predicate symbols and constants. Relational MCSs generalize MCSs, rather than restricting them, since one can take the first-order sublanguage to be empty. However, they allow bridge rules with actual first-order variables, instead of seeing such rules simply as meta-level notation for the (potentially very large) set of all their closed instances. This is useful namely when there is information flow between logic-based systems, as a single rule can "transport" all instances of a predicate from one context to another.

Most of the examples of MCSs presented so far were designed to illustrate the potential of such systems and their variants, but to our knowledge there has not been much effort in the development of general systematic techniques to write MCSs. This is the main achievement of this paper: we propose generic mechanisms, in the form of general-purpose design patterns, that achieve specific forms of interaction between contexts within an MCS in a principled way – e.g. extending a context by means of a definition in the language of another context, giving closed-world semantics for particular predicates in a context with open-world semantics, or reasoning within the merge of two contexts while keeping them separate. The study of these design patterns not only facilitates the development of future MCSs, but also suggests new ways in which their language can be extended. Our departure point was the study of design patterns for multi dl-programs [10] – a generalization of dl-programs [13] with multiple description logic knowledge bases –, which can be seen as a subclass of MCSs by means of a systematic translation [9]. The present study is however much more general than the combination of the work in those two publications.

The paper is organized as follows. Section 2 summarizes previous research relevant to this work. Section 3 recalls the formal definition of relational MCS and introduces an elementary communication pattern for MCSs. Section 4 discusses more general interaction patterns, and Section 5 explores particular applications to MCSs using ontologies. Section 6 discusses future directions for this work.

2 Related work

An emblematic example of best practices in program development are software design patterns, introduced in the mid-nineties by the Gang of Four, whose work [18] paved the way for important advances in software quality; presently, many valuable experienced designers' "best practices" are not only published but effectively used by the software development community. From very basic, abstract, patterns that can be used as building blocks of several more complex ones, to business-specific patterns and frameworks, dozens of design patterns have been proposed [15, 17, 25], establishing a "common language" between development teams that substantially enriches their communication, and hence the whole design process.

Although most of the work around design patterns has been focused in the object-oriented paradigm, several of these patterns are fundamental enough to be independent of the modeling and programming paradigms used. Thus, effort has also been made in adapting some of these best practices to other paradigms and in finding new paradigm-specific patterns [2, 20, 33]. In this line, we contributed to the study of best practices in the field of systems that can access several knowledge bases by identifying several patterns for Mdl-programs [10] – a powerful and expressive formalism to join description logics with rules, connecting a logic program with description logic knowledge bases by means of special dl-atoms that allow information flow [9], generalizing the original dl-programs [13].

Multi-context systems [3] (MCSs) are more general than Mdl-programs and were originally designed to bring together characteristics of both heterogeneous monotonic [21, 27] and homogeneous non-monotonic systems [6, 31], capitalizing on both worlds. They are heterogeneous non-monotonic systems whose components (called *contexts*) are knowledge bases that can be expressed in different logics (e.g., a theory in classical logic, a description logic knowledge base, a set of modal or temporal logic formulas, or a non-monotonic formalism such as a logic program under answer set semantics, or a set of default logic rules). Unlike Mdl-programs, the communication between the components is not centralized, but rather distributed among them via sets of (non-monotonic) *bridge rules*.

Since they were originally proposed, several variations of MCSs have been studied that add to their potential fields of application. Examples are managed MCS [4], whose bridge rules allow arbitrary operations (e.g. deletion or revision operators) on context knowledge bases to be freely defined; relational MCSs [16], which introduce variables and aggregate expressions in bridge rules, extending the semantics of MCSs accordingly; or dynamic MCSs [11], designed to cope with situations where knowledge sources and their contents may change over time and are not known *a priori*. We will work within relational MCSs (defined formally in the next section), and discuss a possible generalization of dynamic MCSs at the end of Section 4.

There are other formalisms to combine different reasoning systems. Hex-programs [14], which generalize dl-programs along a different direction, are higher-order logic programs with external atoms, and they are also heterogeneous since these external atoms may pose queries to systems using different languages. A homogenous approach to combining systems is exemplified by hybrid MKNF knowledge bases [29], which however are not modular. (Partial) translations between these formalisms are an important tool to compare their expressive power and to allow transfer of technology from one formalism into another. Thus, hybrid MKNF knowledge bases can be translated into MCSs [22], providing a way for agents to reason with the former without the need for specialized Hybrid MKNF reasoners. In turn, MCSs and Hex-programs are essentially incomparable [14]. Mdl-programs are trivially embedded in Hex-programs [14], and they can be faithfully translated in a not-so-trivial fashion into MCSs [4, 9].

Yet another way of combining reasoning systems is ontology mediation, which intends to facilitate the interoperability of different ontologies, namely by allowing exchange of instance data through the identification of alignments or the merging of overlapping ontologies. An *alignment* between two distinct ontologies establishes relationships between pairs of entities, one from each ontology. These relationships are then made concrete in the form of ontology mappings, with some tools [7, 12] resorting to “bridge axioms” or even an ontology of generic bridges [26] whose instances constitute concrete bridges defining mappings between the original ontologies. Alignments are also sometimes used as a first step towards defining a single ontology that is a merged version of the original ontologies. However, merging ontologies raises the problem of solving the inconsistencies or incoherences that might arise, which are difficult problems for which several distinct theoretic approaches have been proposed [8], and much effort has been put on the development of tools to assist with ontology merging, see e.g. [24].

Ontology alignment patterns [32] help designers to identify alignments by looking at common patterns of ontology mismatches. Both these and the definition of an ontology of generic mappings are complementary to the construction in Section 5, which translates a previously identified alignment into MCS bridge rules that interconnect two contexts and show how to emulate partial ontology merging within an MCS, given an alignment. The patterns in that section focus on communication and interaction between the ontologies, seen as components of an MCS, and not on their construction and architecture; as such, they are a complement of, rather than an alternative to, ontology design patterns [19].

3 Information flow in relational multi-context systems

We begin this section with a quick summary of the notion of relational multi-context system [16].

A *relational logic* L is a quadruple $\langle \mathsf{KB}_L, \mathsf{BS}_L, \mathsf{ACC}_L, \Sigma_L \rangle$, where KB_L is the set of well-formed logic bases of L, BS_L is a set of possible belief sets, $\mathsf{ACC}_L : \mathsf{KB}_L \to 2^{\mathsf{BS}_L}$ is a function assigning to each knowledge base a set of acceptable sets of beliefs, and Σ_L is a signature consisting of sets P_L^{KB} and P_L^{BS} of predicate names (with associated arity) and a universe U_L of object constants, such that $U_L \cap (P_L^{\mathsf{KB}} \cup P_L^{\mathsf{BS}}) = \emptyset$. The first-order signature Σ_L generates a sublanguage of L, in the sense that $p(c_1, \ldots, c_k)$ must be an element of some knowledge base (resp. belief set), if $p \in P_L^{\mathsf{KB}}$ (resp. $p \in P_L^{\mathsf{BS}}$) has arity k and $c_1, \ldots, c_k \in U_L$. These elements are called *relational ground elements*, while the remaining elements of knowledge bases or belief sets are called *ordinary*.[1]

Let $\mathfrak{I}$ be a finite set of indices, $\{L_i\}_{i \in \mathfrak{I}}$ be a set of relational logics, and V be a set of (first-order) variables distinct from predicate and constant names in any L_i. A *relational element* of L_i has the form $p(t_1, \ldots, t_k)$ where $p \in P_{L_i}^{\mathsf{KB}} \cup P_{L_i}^{\mathsf{BS}}$ has arity k and each t_j is a term from $V \cup U_{L_i}$, for $1 \leq j \leq k$. A *relational k-bridge rule* over $\{L_i\}_{i \in \mathfrak{I}}$ and V is a rule of the form

$$(k : s) \leftarrow (c_1 : p_1), \ldots, (c_q : p_q), \mathsf{not}(c_{q+1} : p_{q+1}), \ldots, \mathsf{not}(c_m : p_m) \qquad (1)$$

such that $k, c_i \in \mathfrak{I}$, s is an ordinary or a relational knowledge base element of L_k and $p_1, \ldots, p_m$ are ordinary or relational beliefs of L_{c_i}.

A *relational multi-context system* is a collection $M = \{C_i\}_{i \in \mathfrak{I}}$ of contexts $C_i = \langle L_i, \mathsf{kb}_i, \mathsf{br}_i, D_i \rangle$, where L_i is a relational logic, $\mathsf{kb}_i \in \mathsf{KB}_L$ is a knowledge base, br_i is a set of relational i-bridge rules, and D_i is a set of import domains $D_{i,j}$, with $j \in \mathfrak{I}$, such that $D_{i,j} \subseteq U_j$. Unless otherwise stated, $D_{i,j}$ is always assumed to be the finite domain consisting of the object constants appearing in kb_j or in the head of a relational bridge rule in br_j.

[1] This notion generalizes that of logic in a general multi-context system, where all elements are ordinary: just take $P_L^{\mathsf{KB}} = P_L^{\mathsf{BS}} = U_L = \emptyset$.

The semantics of relational MCSs is defined in terms of ground instances of bridge rules: the instances obtained from each rule $r \in \mathsf{br}_i$ by uniform substitution of each variable X in r by a constant in $\bigcap D_{i,j}$, with j ranging over the indices of the contexts to which queries containing X are made in r. A *belief state* for M is a collection $S = \{S_i\}_{i\in\mathfrak{I}}$ where $S_i \in \mathsf{BS}_i$ for each $i \in \mathfrak{I}$. The bridge rule (1) is *applicable* w.r.t. belief state S if $p_i \in S_{c_i}$ for $1 \leq i \leq q$ and $p_i \notin S_{c_i}$ for $q < i \leq m$. The set of the heads of all applicable bridge rules of context C_i w.r.t. S is denoted by $\mathsf{app}_i(S)$. An *equilibrium* is a belief state S such that $S_i \in \mathsf{ACC}_i(\mathsf{kb}_i \cup \mathsf{app}_i(S))$. Particular types of equilibria (minimal, grounded, well-founded) that were originally defined for multi-context systems [3] transfer to relational MCSs, but we will not discuss them here.

From this point onwards we will only consider relational MCSs, and omit the adjective "relational" for brevity. The discussion below takes place within the setting of an MCS $M = \{C_i\}_{i\in\mathfrak{I}}$ unless otherwise stated.

The basic communication structure of MCSs can be embodied in a very simple design pattern. Although not very interesting in itself, it is useful as a building block for more elaborate patterns, and we can state and prove its soundness.

Pattern *Observer*.

Problem. There is a predicate name $p \in P_i^{\mathsf{KB}}$ whose semantics should include all instances of predicate names $p_j \in P_{\varphi(j)}^{\mathsf{KB}}$, with $1 \leq j \leq \ell$ and $\varphi(j) \in \mathfrak{I}$, of the same arity.

Solution. Add the bridge rules $(i : p(\boldsymbol{X})) \leftarrow (\varphi(j) : p_j(\boldsymbol{X}))$ to br_i, with $\boldsymbol{X} = X_1, \ldots, X_k$ and $1 \leq j \leq \ell$.

Proposition 1. *Let $M = \{C_i\}_{i\in\mathfrak{I}}$ be an MCS such that $\mathsf{kb} \subseteq \mathsf{ACC}_i(\mathsf{kb})$ for every $\mathsf{kb} \in \mathsf{KB}_i$, and let $p \in P_i^{\mathsf{KB}}$ be defined from $p_j \in P_{\varphi(j)}^{\mathsf{KB}}$ for $j = 1, \ldots, \ell$ by application of **Observer**. Let $S = \{S_i\}_{i\in\mathfrak{I}}$ be an equilibrium for M. For each j, if $p_j(t) \in S_{\varphi(j)}$ for some t, then $p(t) \in S_i$.*

Proof. By definition of equilibrium, if $p_j(t) \in S_{\varphi(j)}$ then $p(t) \in \mathsf{app}_i(S)$ and the thesis follows from the definition of equilibrium and the hypothesis on ACC_i. $\square$

4 Extending expressiveness of contexts

An MCS's information flow capabilities can be applied to extend the language of one context using syntactic means available in another. As a simple example, suppose that we want to define the transitive closure of a binary relation in a context that has no primitives for this. At the semantic level, this can be achieved for named individuals by means of an auxiliary context that can define transitive closures.

We introduce two patterns to deal with this situation; although the first one is a particular case of the second, it is important enough to discuss it on its own.

Pattern *Fixpoint definition*.

Problem. In context C_i we want to define a predicate p from other predicates by means of a logic program.

Solution. Create a new logic programming context C_θ, i.e. a context such that $\mathsf{ACC}_\theta(\mathsf{kb})$ contains only the minimal model of kb over the constants in U_i, and take $D_{i,\theta} = D_{\theta,i} = D_{i,i}$.
Apply ***Observer*** to import from C_i to C_θ all instances of the predicates necessary to define p.
Take kb_θ to be the definition of p.
Apply ***Observer*** to export p from C_θ to C_i.

Proposition 2. *Let predicate p be defined in context C_θ by application of* ***Fixpoint definition*** *and $S = \{S_j\}_{j\in\mathfrak{J}}$ be an equilibrium of the corresponding MCS. Define I to be the restriction of S_i to the Herbrand base of kb_θ (with constants in U_i). Then $S_\theta \subseteq I$.*

Proof. By soundness of ***Observer***, $\mathsf{app}_\theta(S)$ coincides with I except on atoms built from predicate symbol p. By definition of equilibrium, $S_\theta = \mathsf{ACC}_\theta(\mathsf{kb}_\theta \cup \mathsf{app}_\theta(S))$. But since kb_θ only contains the definition of p, S_θ is exactly $\mathsf{app}_\theta(S)$ together with some atoms of the form $p(t)$. By soundness of ***Observer*** all these atoms are in I. □

In particular, this pattern allows us to view deductive databases as multi-context systems – context C_i is the database, context C_θ is the view, and the bridge rules connect them.

In general, it can happen that I contains more information about p; this can be avoided by applying ***Observer*** to both p and $\neg p$ in the last step, but this can easily lead to inconsistency if C_i proves some $p(t)$ that is not derived by C_θ.

There is an important aspect of this construction: it only works at the level of the instances – we are not able to reason abstractly about properties of the defined concepts. In particular, individuals outside the import domain are never "carried over" by bridge rules. This is a necessary evil – otherwise, one would easily get undecidability of reasoning in the resulting MCS.

Example 1. Let C_1 be a context for a decidable fragment of first-order logic where there are binary predicates R, Rt and S, $\mathsf{ACC}_1(\mathsf{kb})$ is the set of logical consequences of kb, and kb_1 contains the axiom $\forall x, y(\mathsf{Rt}(x, y) \rightarrow \mathsf{S}(x, y))$ together with some instances of R (but none of Rt). The goal is to have Rt be the transitive closure of R, but this is not first-order definable.

Application of ***Fixpoint definition*** defines a logic programming context C_2, where kb_2 defines Rt as the transitive closure of R in the usual way, and contains no other rules. Then we add the bridge rules

$$(2 : \mathsf{R}(X, Y)) \leftarrow (1 : \mathsf{R}(X, Y))$$
$$(1 : \mathsf{Rt}(X, Y)) \leftarrow (2 : \mathsf{Rt}(X, Y))$$

to the resulting MCS. In this way, in every equilibrium $\{S_1, S_2\}$ of $\{C_1, C_2\}$ the semantics of Rt in S_1 will coincide with the transitive closure of R in S_1 on named individuals.

However, S_1 does not necessarily satisfy $\forall x, y(\mathsf{R}(x, y) \rightarrow \mathsf{S}(x, y))$: it can happen that $\mathsf{R}(c_1, c_2)$ holds for individuals c_1 and c_2 that are not interpretations of constants in C_1's (syntactic) domain, and the semantics of the bridge rules can not guarantee that $\mathsf{Rt}(c_1, c_2)$, and hence $\mathsf{S}(c_1, c_2)$, holds.

Despite this apparent limitation, this construction works very nicely if C_1 does not allow individuals outside the import domain. This is namely the case if C_1 is a relational or deductive database, or another logic program.

As a generalization of this mechanism, we consider the more encompassing problem of defining a predicate in one context by means of a construct that is only available in other contexts. Typical examples of contexts that could benefit from such additional expressiveness include: description logic contexts, where the available concept/role constructors are restricted to guarantee decidability and complexity bounds on reasoning; relational databases, where no definitional mechanisms exist; or impredicative definitions in first-order contexts. We can achieve this by means of a similar construction: export the instances of the predicates required for the definition into a context that possesses the required ability, write the definition in that context, and import the instances of the defined predicate back into the original context.

Pattern *External definition*.

Problem. In context C_i, we want to define a predicate p by means of a construct that is only available in context C_j.

Solution. Extend $D_{i,j}$ and $D_{j,i}$ with $D_{j,j}$.
Apply ***Observer*** to import all instances of the necessary predicates [and their (default) negations] from C_i to C_j.
Define p in kb_j.
Apply ***Observer*** to export p [and $\neg p$] from C_j to C_i.

There is some freedom regarding whether negations of predicates should be observed; this will depend on the particular application. The soundness of the pattern is proven similarly to the previous case.

Proposition 3. *Let predicate p be defined in context C_i by application of **External definition** and S be an equilibrium of the corresponding MCS. Define I and J to be the restrictions of S_i to the language of C_j and of S_j to the language of C_i, respectively. Then $p(t) \in I$ whenever $p(t) \in J$, with the converse implication holding if all negations are also being observed.*

Both patterns presented in this section fit well with terminological knowledge bases, where concepts are defined in terms of other concepts whose definitions (or instances) may be provided by an external entity.

Another important concern when designing systems is that of querying a context or group of contexts subject to variation minimizing the necessary changes to the contexts querying them. This variation can happen either because that context's contents are expected to change often, or because one does not want to know explicitly which context is being queried when writing bridge rules. (A concrete example will be presented in the next section.) This encapsulation can be achieved by means of the following pattern.

Pattern *Group encapsulation*.

Problem. There are contexts $C_1, \ldots, C_k$ that should be encapsulated, in the sense that other contexts do not include direct queries of the form $(i : p)$ in the bodies of their bridge rules, for $i = 1, \ldots, k$.

Solution. Define functions $\sigma_i : \Sigma_i \to \Sigma_I$ and create a new interface context C_I with $U_I = \bigcup_{i=1}^{k} U_i$, $\mathsf{KB}_I = \left\{ \bigcup_{i=1}^{k} \sigma_i(\mathsf{kb}_i) \mid \mathsf{kb}_i \in \mathsf{KB}_i \right\}$, $\mathsf{kb}_I = \emptyset$, $\mathsf{BS}_I = \mathsf{KB}_I$, $\mathsf{ACC}_I(\mathsf{kb}) = \{\mathsf{kb}\}$, and $D_{I,i} = U_i$ for $i = 1, \ldots, k$.
For every relational symbol $p \in \Sigma_i$, apply ***Observer*** to make $\sigma_i(p)$ in C_I an observer of p.
In every other context, instead of writing $(i : p)$ in the body of a bridge rule, write $(I : \sigma_i(p))$.

By not requiring σ to be an injection, this pattern generalizes ***Observer***.

Proposition 4. *Let M be an MCS where there is an interface context C_I defined by application of **Group encapsulation**. Define M' by removing C_I from M and replacing every bridge rule r with all rules obtained from r by replacing each query $(I : q)$ with a query $(i : p)$ for which $\sigma_i(p) = q$. Then:*

1. If S is an equilibrium of M, then $S_I = \bigcup_{i=1}^{k} \{\sigma_i(p)(t) \mid p(t) \in S_i\}$.
2. S is an equilibrium of M iff $S \setminus S_I$ is an equilibrium of M'.

Proof.

1. The converse inclusion follows by soundness of ***Observer***; the direct inclusion is a consequence of the definition of ACC_I and kb_I.
2. Consequence of the soundness of ***Observer***.

This pattern can be made more general by also considering queries of the form $\mathsf{not}(i : p)$, but we will not discuss this general case here.

Removing the restriction $\mathsf{kb}_I = \emptyset$ we obtain a more powerful design pattern where the interface context is allowed to implement algorithms to decide which contexts to query on what. A more interesting possibility would be to allow a limited form of second-order bridge rules, so that other contexts can query C_I and use the result to know which context to query on which relational symbol.

This kind of approach has been tackled in [22], but the second-order notation therein is interpreted as an abbreviation for all its closed instances – thus solving the presentation problem, but not the practical one. On the other hand, higher-order variables in bridge rules are considered in the schematic contexts of [11], but within a more general setting where they are used as placeholders for contexts that are not known *a priori* and may change over time.

Our tentative proposal would be to allow higher-order variables in bridge rules, these variables being allowed to serve as predicate names or context identifiers (formally numbers, but in practice these could be URLs), with a requirement that their first occurrence in the body of a rule must be positive and in an argument position. This would allow the implementation of indirection-style techniques, with interface contexts serving as mediators indicating what queries to pose to which contexts.

Pattern *Indirection*.

Problem. We want to protect an MCS from variations in bridge rules that include atoms where both the context being queried and the predicate in the query may change with time.

Solution. Create an interface context C_I that implements the algorithm for deciding which contexts should be queried and what the predicate names in actual queries should be.
In every bridge rule with expectable variations, include a query to C_I whose answer provides all required information, and use the result from that query in the subsequent literals in the body of the rule.

The proposal of having higher-order variables in bridge rules as first-class citizens would allow us to have the best of both worlds: the number of actual rules would be kept small, and the configuration algorithm of [11] can be seen as a particular implementation of the interface context. We will return to this issue at the end of the next section.

5 Applications to ontology manipulation

In this section we develop specific mechanisms to deal with MCSs that contain ontologies as contexts. Due to their open-world semantics, this kind of knowledge bases brings specific challenges. In this work, we consider an ontology to be a particular knowledge base whose underlying logic is a description logic.

Definition 1. *An ontology $\mathcal{O}$ expressed in a description logic $\mathcal{L}$ induces the context* $\mathsf{Ctx}(\mathcal{O}) = \langle L, \mathcal{O}, \emptyset, U_\mathcal{L}\rangle$*, with* $L = \langle \mathsf{KB}_\mathcal{L}, \mathsf{BS}_\mathcal{L}, \mathsf{ACC}_\mathcal{L}, \Sigma_\mathcal{L}\rangle$ *defined as follows.*

- $\mathsf{KB}_\mathcal{L}$ *is the set of all well-formed knowledge bases of* $\mathcal{L}$*.*
- $\mathsf{BS}_\mathcal{L}$ *is the set of all sets of literals in the language of* $\mathcal{L}$*.*
- $\mathsf{ACC}_\mathcal{L}(\mathsf{kb})$ *is the singleton set containing the set of* kb*'s known consequences (positive and negative).*
- $\Sigma_\mathcal{L}$ *is the first-order signature underlying* $\mathcal{L}$*.*

The belief sets (the elements of $\mathsf{BS}_\mathcal{L}$) do not need to be categorical: they may contain neither $\mathsf{C}(\mathsf{a})$ nor $\neg\mathsf{C}(\mathsf{a})$ for a particular concept C and individual a. This is what gives description logic knowledge bases their typical open-world semantics. For this reason, the only element of $\mathsf{ACC}_\mathcal{L}(\mathsf{kb})$ may not be a model of kb. This is in contrast with [3], where $\mathsf{ACC}_\mathcal{L}(\mathsf{kb})$ contains all (first-order) models of kb. We will return to this issue in Example 2 below.

Default reasoning. In the framework of dl-programs, it has been shown [13] that default rules can be implemented in a systematic way. The construction proposed by those authors can be simplified in the framework of multi-context systems.

A default rule has the form $\dfrac{\alpha_1,\ldots,\alpha_k : \beta_1,\ldots,\beta_n}{\gamma}$, where α_i, β_j and γ are literals for all i, j, with intended semantics that if, for some instantiation θ of the free variables in the rule, all $\alpha_i\theta$ hold and it is consistent to assume that all $\beta_j\theta$ hold, then $\gamma\theta$ is inferred. Several semantics for default rules have been proposed [1], namely Reiter's original semantics [30] based on *extensions* – theories that are fixpoints w.r.t. the default rules.

Pattern *Default rule*.

Problem. Context C_i should include the default rule $\dfrac{\alpha_1,\ldots,\alpha_k : \beta_1,\ldots,\beta_n}{\gamma}$.

Solution. Include the bridge rule $(i : \gamma) \leftarrow (i : \alpha_1),\ldots,(i : \alpha_k), \mathsf{not}(i : \beta_1),\ldots,\mathsf{not}(i : \beta_n)$ in br_i.

The result below makes the correspondence with the standard default semantics precise, considering *minimal* equilibria – equilibria whose belief sets are not proper supersets of any other equilibria.[2]

Proposition 5. *Let $\mathcal{O}$ be an ontology and Γ be a set of default rules in the language of $\mathcal{O}$. Let M be the MCS with a single context $\mathsf{Ctx}(\mathcal{O})$ and bridge rules obtained by applying* ***Default rule*** *to the rules in Γ. Then S is a minimal equilibrium of M iff S is an extension of $\mathcal{O}$ and Γ.*

Proof. We use Reiter's characterization of extensions [30]. For a set of formulas F, let $\mathcal{E}(F,0) = \mathcal{O}$ and $\mathcal{E}(F,i+1)$ be the theory generated from $\mathcal{E}(F,i)$ and all γ such that $\dfrac{\alpha_1,\ldots,\alpha_k : \beta_1,\ldots,\beta_n}{\gamma} \in \Gamma$, $\alpha_j \in \mathcal{E}(F,i)$ for $1 \leq j \leq k$, and $\neg\beta_j \notin F$ for $1 \leq j \leq n$. Let $\mathcal{E}_F = \bigcup_{i=0}^{+\infty} \mathcal{E}(F,i)$. Then E is an extension of $\mathcal{O}$ and Γ iff $E = \mathcal{E}_E$.

In particular, for any extension E of $\mathcal{O}$ and Γ, $E = \mathsf{ACC}_{\mathcal{L}}(E \cup \mathsf{app}_E(E))$, whence E is an equilibrium of M. Conversely, if S is an equilibrium of M, then by induction $\mathcal{E}(S,i) \subseteq S$: for $i = 0$ this is trivial; for $i+1$ assume that $\mathcal{E}(S,i) \subseteq S$ and note that $\mathcal{E}(S,i+1)$ is then derived from $\mathcal{E}(S,i)$ and the heads of rules that are applicable in $\mathcal{E}(S,i)$. Hence $\mathcal{E}_S \subseteq S$.

Then: if S is an equilibrium of M, then $\mathcal{E}_S \subseteq S$ is also an equilibrium of M that is simultaneously an extension of $\mathcal{O}$ and Γ; if S is a minimal equilibrium, then necessarily $\mathcal{E}_S = S$, yielding the thesis. Conversely, if E is an extension of $\mathcal{O}$ and Γ, then it is an equilibrium of M, and since extensions are minimal w.r.t. set inclusion it must be a minimal equilibrium. $\square$

This result can be made a bit stronger; under some conditions, which often arise in practice, only minimal equilibria exist.

[2] In this setting, this simplified definition of minimal equilibria is equivalent to the original one in [3].

Corollary 1. *Let $\mathcal{O}$, Γ and M be as in Proposition 5 and suppose that, for every extension E and rule $\dfrac{\alpha_1,\ldots,\alpha_k:\beta_1,\ldots,\beta_n}{\gamma}\in\Gamma$, $\alpha_i\in E$ iff α_i is a consequence of $\mathcal{O}$. Then every equilibrium of M is an extension of $\mathcal{O}$ and Γ.*

Proof. Under the hypothesis, $\alpha_j \in E_i$ iff $\alpha_j \in E$, and the thesis follows. □

In particular, if the rules in Γ are *prerequisite free* [5] (i.e. $k = 0$), then every equilibrium of M corresponds to an extension of $\mathcal{O}$ and Γ, and conversely. This is interesting in practice, as it corresponds to many useful applications such as the modeling of closed-world reasoning by means of default rules [5]. For this correspondence to hold, however, it is essential that $\mathsf{Ctx}(\mathcal{O})$ be defined as above, and not by having the usual models-as-belief-sets construction of [3].

Example 2. Suppose that $\mathcal{O}$ is the ontology consisting of the single formula $\mathsf{C}(\mathsf{a})\sqcup\mathsf{C}(\mathsf{b})$. Then $\mathcal{O}$'s models must contain at least one of $\mathsf{C}(\mathsf{a})$ or $\mathsf{C}(\mathsf{b})$. Since none of these is guaranteed to hold in all models, $\mathsf{ACC}_{\mathcal{L}}(\mathcal{O}) = \emptyset$. Adding closed-world semantics to C, by means of the translated default rule $(1:\neg\mathsf{C}(X)) \leftarrow \mathsf{not}(1:\mathsf{C}(X))$, yields two possible equilibria, corresponding to the two extensions of the corresponding default rule: $\{\mathsf{C}(\mathsf{a}),\neg\mathsf{C}(\mathsf{b})\}$ and $\{\mathsf{C}(\mathsf{b}),\neg\mathsf{C}(\mathsf{a})\}$.

With the approach from [3], $\mathsf{ACC}_{\mathcal{L}}(\mathcal{O})$ contains the three models $\{\mathsf{C}(\mathsf{a}),\neg\mathsf{C}(\mathsf{b})\}$, $\{\mathsf{C}(\mathsf{b}),\neg\mathsf{C}(\mathsf{a})\}$ and $\{\mathsf{C}(\mathsf{a}),\mathsf{C}(\mathsf{b})\}$. The bridge rule above has no effect, and adding it to the corresponding MCS still yields three equilibria, one of which does not correspond to an extension of $\mathcal{O}$ and that rule.

The pattern ***Default rule*** generalizes the notion of default rule to multi-context systems that are not generated from an ontology. Furthermore, we can remove the requirement that all literals in the rule come from the same context and encode more general default rules. By the results above, we can see minimal equilibria for MCSs with applications of this pattern as generalized default extensions, obtaining a systematic way to approximate closed-world reasoning in a standard way.

In order to obtain *true* closed-world reasoning (in the sense that e.g. the MCS in Example 2 would be inconsistent, as $\mathcal{O}$ is inconsistent with the closed-world assumption) one could define ACC_i as a binary operator, separating the original belief state from the conclusions derived from the application of bridge rules and allowing them to be treated differently. We are currently studying the impact of this change in the theory of MCSs.

Working with alignments. Another, seemingly unrelated, problem that occurs quite often in practice is that of reasoning within the merge of two ontologies, given an alignment, without actually constructing the merged ontology.

An *alignment* between two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ is a set $\mathcal{A}$ of atoms $t(P,Q)$ where P is a concept (or role) from $\mathcal{O}_1$, Q is a concept (resp. role) from $\mathcal{O}_2$, and $t \in \{\mathsf{subsumed},\mathsf{subsumes},\mathsf{equivalent},\mathsf{disjoint}\}$ (see [32]).

Definition 2. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two ontologies and $\mathcal{A}$ be an alignment between them. The MCS* induced *by $\mathcal{A}$ is $M(\mathcal{O}_1,\mathcal{O}_2,\mathcal{A})$, containing $\mathsf{Ctx}(\mathcal{O}_1)$ and $\mathsf{Ctx}(\mathcal{O}_2)$ with the following bridge rules:*

– *for each triple* $\mathsf{subsumed}(P,Q) \in \mathcal{A}$,

$$(2 : Q(X)) \leftarrow (1 : P(X)) \tag{2}$$
$$(1 : \neg P(X)) \leftarrow (2 : \neg Q(X)) \tag{3}$$

if P *and* Q *are concepts, or their binary counterparts, if they are roles;*

– *for each triple* $\mathsf{disjoint}(P,Q) \in \mathcal{A}$,

$$(2 : \neg Q(X)) \leftarrow (1 : P(X)) \tag{4}$$
$$(1 : \neg P(X)) \leftarrow (2 : Q(X)) \tag{5}$$

if P *and* Q *are concepts, or their binary counterparts, if they are roles.*

The triple $\mathsf{subsumes}(P,Q)$ *is treated as* $\mathsf{subsumed}(Q,P)$*, with the appropriate context changes, and* $\mathsf{equivalent}(P,Q)$ *is seen as the conjunction of* $\mathsf{subsumed}(P,Q)$ *and* $\mathsf{subsumed}(Q,P)$*.*

Again this achieves a merge of $\mathcal{O}_1$ and $\mathcal{O}_2$ at the level of the individuals in the import domain – we cannot reason e.g. about the potential subsumption of a concept from $\mathcal{O}_1$ by a concept from $\mathcal{O}_2$. Still, this construction is useful as it avoids the extra step of constructing a new ontology.

There is a more practical aspect of this approach that we can ameliorate: when querying $M(\mathcal{O}_1,\mathcal{O}_2,\mathcal{A})$, one must know where the concept or role in the query originates from. This issue can be bypassed by applying ***Group Encapsulation*** to hide the two contexts in this MCS.

Pattern *Alignment*.

Problem. We want to reason about the instances in the merge of two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ w.r.t. a given (consistent) alignment $\mathcal{A}$, without building the merged ontology.
Solution. Apply ***Group Encapsulation*** to $M(\mathcal{O}_1,\mathcal{O}_2,\mathcal{A})$.

This design pattern assumes that $\mathcal{A}$ is consistent; however, it may happen that it is not (yet) guaranteed to be consistent. In order to avoid possible inconsistencies at the level of the instances, we may use the more robust technique from [28], taking the maximal consistent merge of $\mathcal{O}_1$ and $\mathcal{O}_2$ by writing the alignment triples as default rules. This translates to constructing $M'(\mathcal{O}_1,\mathcal{O}_2,\mathcal{A})$ as $M(\mathcal{O}_1,\mathcal{O}_2,\mathcal{A})$ but replacing the bridge rules with those obtained by ***Default Rule***, protecting the context from the introduction of (explicit) inconsistencies. For example, if C and D are concepts, then $\mathsf{subsumed}(C,D)$ would yield

$$(2 : D(X)) \leftarrow (1 : C(X)), \mathsf{not}\ (2 : \neg D(X))$$
$$(1 : \neg C(X)) \leftarrow (2 : \neg D(X)), \mathsf{not}\ (1 : C(X))$$

Example 3. Consider a very simple case where $\mathcal{O}_1$ has the two instance axioms $\mathsf{C}(\mathsf{a})$ and $\mathsf{C}(\mathsf{b})$, $\mathcal{O}_2$ only has the axiom $\neg\mathsf{D}(\mathsf{a}) \sqcup \neg\mathsf{D}(\mathsf{b})$, and $\mathcal{A}$ contains $\mathsf{subsumed}(\mathsf{C},\mathsf{D})$. Note that $\mathcal{A}$ is inconsistent with $\mathcal{O}_1$ and $\mathcal{O}_2$. Then $M'(\mathcal{O}_1,\mathcal{O}_2,\mathcal{A})$

has two distinct equilibria, namely $S^1 = \langle\{\mathsf{C}(\mathsf{a}), \mathsf{C}(\mathsf{b})\}, \{\mathsf{D}(\mathsf{a}), \neg\mathsf{D}(\mathsf{b})\}\rangle$ and $S^2 = \langle\{\mathsf{C}(\mathsf{a}), \mathsf{C}(\mathsf{b})\}, \{\neg\mathsf{D}(\mathsf{a}), \mathsf{D}(\mathsf{b})\}\rangle$. In both, the semantics of D is maximal (it includes as many instances of C as it may consistently do). As expected, none of these equilibria satisfies the alignment axiom $\mathsf{C} \sqsubseteq \mathsf{D}$.

There is a drawback to this construction: the high number of bridge rules required, which grows with the number of concepts and roles in $\mathcal{O}_1$ and $\mathcal{O}_2$. It would be useful to be able to write these bridge rules in a second-order language, e.g. rule (2) would become

$$(2 : D(X)) \leftarrow (0 : \mathsf{subsumes}(C, D)), (1 : C(X)) \tag{6}$$

where context C_0 is simply $\mathcal{A}$. The interesting aspect is that context C_0 can actually be seen as a relational (first-order) context – it is the usage of its "constants" as predicate names in the bridge rules that gives them a higher-order nature. We are currently working on developing a formal theory of MCSs with higher-order rules.

6 Conclusions

In this paper we addressed several issues related to the flow of information between the several components of a relational multi-context system, presenting general-purpose design patterns that systematize the constructs supporting this communication.

Due to the specific semantics of bridge rules, these constructions only affect the individuals in the import domains of the contexts where new predicates are defined. This apparent limitation is however essential to avoid fundamental inconsistency and undecidability problems: the main advantage of several of these design patterns is precisely that they allow one to mimic extending the expressiveness of a context for one particular definition in a way that, in its full generality, would render the context inconsistent or undecidable.

We also study particular constructions adapted to working with ontologies, introducing a new definition of the context generated from an ontology in a way that is fundamentally different from what had been previously suggested [3]. This new definition truly captures the nature of the open-world vs closed-world semantics in the form required to allow fine-tuning of the particular interpretation for specific predicates.

The development of these design patterns also suggests the study of syntactic extensions to MCSs: changing the consequence operator to a binary one, allowing different treatment of data from the knowledge base and that inferred from application of bridge rules, in order to have true closed-world reasoning; and higher-order bridge rules, where the result of queries to one context can be used to decide *which* predicates to use on subsequent queries to other contexts, or even *to which* context those queries should be made. The last construct has been used – but in the form of meta-level notation, as an abbreviation for a set of rules – in work by other authors [11, 22]. We are currently working on making these two constructions first-class citizens of multi-context systems, allowing them in the syntax of bridge rules and studying their semantics formally.

References

1. G. Antoniou. A tutorial on default logics. *ACM Computing Surveys*, 31(3):337–359, 1999.
2. S. Antoy and M. Hanus. Functional logic design patterns. In Z. Hu and M. Rodríguez-Artalejo, editors, *FLOPS 2002*, volume 2441 of *LNCS*, pages 67–87. Springer, 2002.
3. G. Brewka and T. Eiter. Equilibria in heterogeneous nonmonotonic multi-context systems. In *AAAI2007*, pages 385–390. AAAI Press, 2007.
4. G. Brewka, T. Eiter, M. Fink, and A. Weinzierl. Managed multi-context systems. In T. Walsh, editor, *IJCAI*, pages 786–791. IJCAI/AAAI, 2011.
5. G. Brewka, I. Niemel, and M. Truszczyński. Nonmonotonic reasoning. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, chapter 6, pages 239–284. Elsevier, 2008.
6. G. Brewka, F. Roelofsen, and L. Serafini. Contextual default reasoning. In M.M. Veloso, editor, *IJCAI2007*, pages 268–273, 2007.
7. J. de Bruijn, M. Ehrig, C. Feier, F. Martíns-Recuerda, F. Scharffe, and M. Weiten. Ontology mediation, merging, and aligning. In J. Davies, R. Studer, and P. Warren, editors, *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. John Wiley & Sons, Ltd, Chichester, UK, 2006.
8. R. Cóbe and R. Wassermann. Ontology merging and conflict resolution: Inconsistency and incoherence solving approaches. In *Workshop on Belief change, Nonmonotonic reasoning and Conflict Resolution (BNC)*, 2012.
9. L. Cruz-Filipe, R. Henriques, and I. Nunes. Description logics, rules and multi-context systems. accepted for publication in *Proceedings of LPAR 2013*, 2013.
10. L. Cruz-Filipe, I. Nunes, and G. Gaspar. Patterns for interfacing between logic programs and multiple ontologies. In Joaquim Filipe and Jan Dietz, editors, *KEOD2013*, pages 58–69. INSTICC, 2013.
11. M. Dao-Tran, T. Eiter, M. Fink, and T. Krennwallner. Dynamic distributed nonmonotonic multi-context systems. In G. Brewka, V. Marek, and M. Truszczynski, editors, *Nonmonotonic Reasoning, Essays Celebrating its 30th Anniversary*, volume 31 of *Studies in Logic*. College Publications, 2011.
12. Dejing Dou, Drew McDermott, and Peishen Qi. Ontology translation by ontology merging and automated reasoning. In *Ontologies for Agents: Theory and Experiences*, pages 73–94. Springer, 2005.
13. T. Eiter, G. Ianni, T. Lukasiewicz, and R. Schindlauer. Well-founded semantics for description logic programs in the semantic Web. *ACM Transactions on Computational Logic*, 12(2), 2011. Article Nr 11.
14. T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In Kaelbling and Saffiotti [23], pages 90–96.
15. T. Erl. *SOA Design Patterns*. Prentice Hall, New York, 2009.
16. M. Fink, L. Ghionna, and A. Weinzierl. Relational information exchange and aggregation in multi-context systems. In J.P. Delgrande and W. Faber, editors, *LPNMR*, volume 6645 of *LNCS*, pages 120–133. Springer, 2011.
17. M. Fowler. *Patterns of Enterprise Application Architecture*. Addison–Wesley, 2002.

18. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison–Wesley, 1995.
19. A. Gangemi and V. Presutti. Ontology design patterns. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 221–243. Springer, 2009. 2nd edition.
20. J. Gibbons. Design patterns as higher-order datatype-generic programs. In R. Hinze, editor, *WGP 2006*, pages 1–12. ACM, 2006.
21. F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics, or: how we can do without modal logics. *Artificial Intelligence*, 65(1):29–70, 1994.
22. M. Homola, M. Knorr, J. Leite, and M. Slota. MKNF knowledge bases in multi-context systems. In M. Fisher, L. van der Torre, M. Dastani, and G. Governatori, editors, *CLIMA*, volume 7486 of *LNCS*, pages 146–162. Springer, 2012.
23. L.P. Kaelbling and A. Saffiotti, editors. *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30–August 5, 2005.* Professional Book Center, 2005.
24. J. Kim, M. Jang, Y. Ha, J. Sohn, and S. Lee. MoA: OWL ontology merging and alignment tool for the semantic web. In M. Ali and F. Esposito, editors, *IEA/AIE2005*, volume 3533 of *LNCS*, pages 722–731. Springer, 2005.
25. C. Larman. *Applying UML and Patterns.* Prentice–Hall, 2004. 3rd Edition.
26. A. Maedche, B. Motik, N. Silva, and R. Volz. MAFRA – a MApping FRAmework for Distributed Ontologies. In A. Gómez-Pérez and V.R. Benjamins, editors, *EKAW2002*, volume 2473 of *LNCS*, pages 235–250. Springer, 2002.
27. J. McCarthy. Notes on formalizing context. In R. Bajcsy, editor, *IJCAI1993*, pages 555–562. Morgan Kaufmann, 1993.
28. T.A. Meyer, K. Lee, and R. Booth. Knowledge integration for description logics. In M.M. Veloso and S. Kambhampati, editors, *AAAI2005*, pages 645–650. AAAI Press / The MIT Press, 2005.
29. B. Motik and R. Rosati. Reconciling description logics and rules. *Journal of the ACM*, 57, June 2010. Article Nr 30.
30. R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
31. F. Roelofsen and L. Serafini. Minimal and absent information in contexts. In Kaelbling and Saffiotti [23], pages 558–563.
32. F. Scharffe, O. Zamazal, and D. Fensel. Ontology alignment design patterns. *Knowledge and Information Systems*, pages 1–28, April 2013.
33. L. Sterling. Patterns for Prolog programming. In A.C. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*, volume 2407 of *LNCS*, pages 374–401. Springer, 2002.