Managed Multi-Context Systems*

Gerhard Brewka† **Thomas Eiter**‡ **Michael Fink**‡ **Antonius Weinzierl**‡

†University of Leipzig
Department of Computer Science
brewka@informatik.uni-leipzig.de

‡Vienna University of Technology
Institute of Information Systems
{eiter,fink,weinzierl}@kr.tuwien.ac.at

Abstract

Multi-context systems (MCS) are a powerful framework for interlinking heterogeneous knowledge sources. They model the flow of information among different reasoning components (called contexts) in a declarative way, using so-called bridge rules, where contexts and bridge rules may be nonmonotonic. We considerably generalize MCS to managed MCS (mMCS): while the original bridge rules can only add information to contexts, our generalization allows arbitrary operations on context knowledge bases to be freely defined, e.g., deletion or revision operators. The paper motivates and introduces the generalized framework and presents several interesting instances. Furthermore, we consider inconsistency management in mMCS and complexity issues.

1 Introduction

Over the last decades research in knowledge representation and, more generally, information technology has produced a large variety of formats and languages for representing knowledge. A wealth of tools and formalisms is now available, including rather basic ones like databases or the more recent triple-stores, and more expressive ones like ontology languages (e.g., description logics), temporal and modal logics, nonmonotonic logics, or logic programs under answer set semantics, to name just a few (see van Harmelen *et al.* [2008]). Several are widely used, a development particularly fueled by the advent and continuous growth of the world wide web.

However, the diversity of formalisms also poses some important challenges: What is the best way to benefit from it? How to build systems that need access to heterogeneous knowledge sources, e.g., in ambient intelligence [Bikakis and Antoniou, 2008]? A standardized, universal knowledge representation language certainly cannot be the solution. There are very good reasons for special purpose representation languages, e.g., domain specific modeling needs and complexity.

Nonmonotonic multi-context systems (MCS) [Brewka and Eiter, 2007] provide a promising way to address the challenges above. The basic idea is to leave the diverse formalisms and knowledge bases (called contexts in this approach for historical reasons [Giunchiglia and Serafini, 1994]) as they are, but to model the necessary information flow among contexts using so-called bridge rules. The latter are similar to logic programming rules (including default negation), but allow access to other contexts in their bodies. This has several advantages: the specification of the information flow is fully declarative; moreover, not only can information be passed from one context to another *as is*; bridge rules also provide means, e.g., for selection, abstraction and conflict resolution between contexts.

Example 1. *Consider a pharmaceutical company producing drugs. A drug effect database C_1 holds information about what is the remedy of an illness caused by certain bacteria, also treatments known to be ineffective are stored. To maximize efficiency, the company wants to know all kinds of illness that can be cured by their drugs. A public health RDF-triple store C_2 is queried on illness caused by bacteria for which C_1 already holds a remedy. Furthermore, probable influence of the drugs on other bacteria is derived which enables more focused clinical trials, i.e., only those effects are tested in the later trial where a likely effect was found. This is realized using an ontology about bacteria C_3 and a third party reasoner to derive likely drug effects on other bacteria. To this end, C_1 may use a view restricted to the necessary information.*

Although MCS are, as we believe, an excellent starting point to address the problems from above, the way they integrate knowledge is still quite limited: if a bridge rule for a context is applicable, then the rule head is simply added to the context's knowledge base (KB). Although this covers the flow of information, it does not capture other operations one may want to perform on context KBs. For instance, rather than simply *adding* a formula ϕ, we may want to *revise* [Peppas, 2008] the KB with ϕ to avoid inconsistency in the context's belief set; and, we may want to modify only certain parts of KB such that ϕ is entailed. In Example 1, an update of C_1's view to C_3 may be wanted (which must be realized inside C_3).

For this reason, we introduce in this paper a new, more general form of MCS where such additional operations on knowledge bases can be freely defined; this is akin to management functionality of database systems. We call the additional component *context manager* and the generalized systems *managed* multi-context systems (mMCS).

The outline of the paper is as follows: after recalling the necessary background on MCS, we introduce the new frame-

*Supported by the Vienna Science and Technology Fund (WWTF) project ICT08-020.

work in Section 3, while Section 4 discusses sample instances. Several aspects of inconsistency management in mMCS are addressed in Section 5 and complexity issues in Section 6.

2 Preliminaries

Multi-Context Systems as defined in Brewka and Eiter [2007] build on an abstract notion of a *logic* L as a triple (KB_L, BS_L, ACC_L), where KB_L is the set of admissible knowledge bases of L, which are sets of KB-elements ("formulas"); BS_L is the set of possible belief sets, whose elements are beliefs; and $ACC_L : KB_L \rightarrow 2^{BS_L}$ is a function describing the semantics of L by assigning each knowledge-base a set of acceptable belief sets.

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 of the form:

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

such that $kb \cup \{s\}$ is an element of KB_{L_i}, $c_\ell \in \{1, \ldots, n\}$, and p_ℓ is element of some belief set of BS_{c_ℓ}, for all $1 \leq \ell \leq m$. For a bridge rule r, we denote by $hd(r)$ the formula s while $body(r)$ denotes the set $\{(c_{\ell_1} : p_{\ell_1}) \mid 1 \leq \ell_1 \leq j\} \cup \{not(c_{\ell_2} : p_{\ell_2}) \mid j < \ell_2 \leq m\}$.

Example 2 (ctd). *In our running example, C_1 is a relational database which contains the information that penicillin remedies pneumonia caused by the bacteria streptococcus pneumoniae and also that azithromycin remedies Legionair's disease caused by legionella pneumophila, effectiveness of both remedies is backed by clinical trials and therefore evident. Also, it contains information that penicillin is ineffective against legionella pneumophila. C_2 is an RDF-triple store containing information that streptococcus pneumoniae causes meningitis and legionella pneumonphila causes atypical pneumonia. Formally, the knowledge bases of C_1, respectively C_2, are:*

$$\begin{aligned} kb_1 = \{&treat(pen, str\_pneu, pneu, evd), \\ &treat(azith, leg\_pneu, leg, evd), \\ &ineff(pen, leg\_pneu)\}, \\ kb_2 = \{&str\_pneu\ rdf{:}causes\ men, \\ &leg\_pneu\ rdf{:}causes\ atyp\_pneu\}. \end{aligned}$$

To incorporate information from C_2, C_1 uses the following bridge rule connecting existing information about effects on bacteria with illness caused by the same bacteria. The incorporated information is further marked as only being likely, but not clinically tested. The set of bridge rules of C_1 is given by:

$$br_1 = \{treat(X, B, I, likely) \leftarrow (1 : treat\ (X, B, \_, \_))\ , \\ (2 : B\ rdf{:}causes\ I)\ .\}.$$

Here we use for readability and succinctness schematic bridge rules with variables (upper case letters and _) which range over associated sets of constants; they stand for all respective instances (obtainable by value substitution).

A belief state $S = (S_1, \ldots, S_n)$ for M consists of belief sets $S_i \in BS_i$, $1 \leq i \leq n$. A bridge rule r of form (1) is applicable wrt. S, denoted by $S \models body(r)$, iff $p_\ell \in S_{c_\ell}$ for $1 \leq \ell \leq j$ and $p_\ell \notin S_{c_\ell}$ for $j < \ell \leq m$. We use $app_i(S) = \{hd(r) \mid r \in br_i \wedge S \models body(r)\}$ to denote the heads of all applicable bridge rules of context C_i wrt. S.

The semantics of an MCS M is then defined in terms of equilibria, where an *equilibrium* is a belief state $(S_1, \ldots, S_n)$ such that $S_i \in ACC_i(kb_i \cup app_i(S))$, $1 \leq i \leq n$.

Example 3 (ctd). *Suppose the possible belief sets in C_1 and C_2 are all sets of facts (over the respective vocabularies) and the acceptable ones for a knowledge base kb are those holding the facts which logically follow from kb under the respective inference closure inf_i, i.e., $ACC_i(kb) = \{inf_i(kb)\}$. In our example we have a single equilibrium $S = (S_1, S_2)$, given by $S_1 = inf_1(kb_1 \cup app_1(S)) = kb_1 \cup \{treat(pen, str\_pneu, men, likely),\ treat(azith, leg\_pneu, atyp\_pneu, likely)\}$ and $S_2 = inf_2(kb_2 \cup app_2(S)) = kb_2$.*

3 Managed Multi-Context Systems

Multi-Context Systems allow us to increase the knowledge base of a context using information from other contexts, but not to operate on it in other ways. Such other operations might be: removal of information; revision with new information, or other complex operations like view-updates of databases; program updates of logic programs; modifications of argumentation frameworks, etc. Notably, those operations are realized by *legacy systems*, but the MCS framework can not cope with this functionality in a principled way.

To enable such functionality with a clear distinction between the knowledge base and additional operations on it, we introduce the managed Multi-Context Systems framework. The latter extends Multi-Context Systems such that contexts come with an additional managing function which evaluates the aforementioned operations, in analogy to the distinction of a database (DB) and a database management system (DBMS).

Example 4 (ctd). *Assume C_1's data is offered to a third party C_4 for deriving further possible drug effects. To that end, a view "eff" is created containing only the information about which drug effects what bacteria, i.e., $(X, B) \in eff$ iff $(X, B, \_, evd) \in treat$, as information derived from C_4 is also returned through this view, it must be updateable.*

C_4 is a logic program accessing the ontology C_3 on bacteriological relations to deduce probable effects on related bacteria. C_1 also holds information on ineffective drugs which should be incorporated into C_4 by means of a logic program update. This allows to deduce probable effects on bacteria except for cases where there is already negative evidence.

To accommodate flexible semantics of contexts, we extend the above notion of "logic" to one which has several semantics to choose from. This allows, e.g., that a logic program is evaluated using well-founded semantics instead of stable semantics based on input from other contexts, or switching from classical to paraconsistent semantics.

Definition 1. *A* logic suite *$LS = (BS_{LS}, KB_{LS}, \mathcal{ACC}_{LS})$ consists of the set BS_{LS} of possible belief sets, the set KB_{LS} of well-formed knowledge-bases, and a nonempty set $\mathcal{ACC}_{LS}$ of possible semantics of LS, i.e, $ACC_{LS} \in \mathcal{ACC}_{LS}$ implies $ACC_{LS} : KB_{LS} \rightarrow 2^{BS_{LS}}$.*

For a logic suite LS, $F_{LS} = \{s \in kb \mid kb \in KB_{LS}\}$ is the set of formulas occurring in its knowledge bases.

Definition 2. *A* management base *is a set of operation names (briefly, operations) OP.*

Intuitively, a management base is the set of commands that can be executed on formulas, e.g., addition of, revision with, etc. a formula. For a logic suite LS and a management base OP, let $F^{OP}_{LS} = \{o(s) \mid o \in OP, s \in F_{LS}\}$ be the set of operational statements that can be built from OP and F_{LS}.

The semantics of such statements is given by a management function. A management function maps a set of operational statements and a knowledge base to pairs of a modified knowledge base and a semantics. It allows to not only add formulas to a context, but to specify any desired operations to be applied on formulas and a context.

Definition 3. *A* management function *over a logic suite LS and a management base OP is a function $mng : 2^{F^{OP}_{LS}} \times KB_{LS} \rightarrow 2^{(KB_{LS} \times \mathcal{ACC}_{LS})} \setminus \{\emptyset\}$.*

Bridge rules for context C_i are now of form (1) as for MCS, but with the head expression s being an operational statement for the management function mng_i of C_i; let OP_i be the management base of C_i, then $s \in F^{OP_i}_{LS_i}$.

Example 5 (ctd). *Regarding C_1, an operation "insert" is introduced and the management function mng_1 is defined such that tuples inserted into the view eff are transformed to treat tuples; formally, $mng_1(O, kb_1) = \{(kb_1 \cup N \cup V, ACC_1)\}$ where $N = \{treat(\vec{t}) \mid insert(treat(\vec{t})) \in O\}$, $V = \{treat(X, B, I, est) \mid insert(eff(X, B)) \in O \wedge treat(\_, B, I, \_) \in kb_1 \cup N\}$ and kb_1 is extended by the following view definition: $eff(X, B) \leftarrow treat(X, \_, B, evd)$.*

For C_4, the operations are $OP_4 = \{upd, add\}$ and mng_4 adds all formulas in add as facts and the resulting program is updated according to formulas in upd by the method given in Alferes et al. *[2002], which avoids inference of conflicting information (see Ex. 7 for details). Thus, mng_4 always selects the stable model semantics of dynamic logic programs.*

A managed MCS (mMCS) is then an MCS where each context additionally comes with a management function and bridge rule heads are operational statements.

Definition 4. *A* managed Multi-Context System *M is a collection $(C_1, \ldots, C_n)$ of managed contexts where, for $1 \leq i \leq n$, each* managed context *C_i is a quintuple $C_i = (LS_i, kb_i, br_i, OP_i, mng_i)$ such that*

- *$LS_i = (BS_{LS_i}, KB_{LS_i}, \mathcal{ACC}_{LS_i})$ is a logic suite,*
- *$kb_i \in KB_{LS_i}$ is a knowledge base,*
- *br_i is a set of bridge rules for C_i,*
- *OP_i is a management base, and*
- *mng_i is a management function over LS_i and OP_i.*

As for ordinary MCS, a belief state $S = (S_1, \ldots, S_n)$ of M is a belief set for every context, i.e., $S_i \in BS_{LS_i}$. Again, by $app_i(S) = \{hd(r) \mid r \in br_i \wedge S \models body(r)\}$ we denote the set of applicable heads of bridge rules, which is a set of operational statements.

The semantics of mMCS is now in terms of equilibria.

Definition 5. *Let $M = (C_1, \ldots, C_n)$ be an mMCS. A belief state $S = (S_1, \ldots, S_n)$ is an equilibrium of M iff for every $1 \leq i \leq n$ there exists some $(kb'_i, ACC_{LS_i}) \in mng_i(app_i(S), kb_i)$ such that $S_i \in ACC_{LS_i}(kb'_i)$.*

Observe that an ordinary context $C_i = (L_i, kb_i, br_i)$ with logic $L_i = (BS_{L_i}, KB_{L_i}, ACC_{L_i})$ can be easily turned into a managed context $C'_i = (LS_i{}', kb_i, br'_i, OP_i, add_i)$ over $LS_i = (BS_{L_i}, KB_{L_i}, \{ACC_{L_i}\})$, where $OP_i = \{add_i\}$ and add_i interprets $add(f)$ as addition of f, i.e., $add_i(O, kb_i) = \{(kb_i \cup \{s \mid add(s) \in O\}, ACC_{L_i})\}$ and $br'_i = \{add(s) \leftarrow body(r) \mid r \in br_i \wedge hd(r) = s\}$. We call C'_i the *management version* of C_i and for convenience we identify both. Managed MCS thus generalize ordinary MCS.

Proposition 1. *Let $M = (C_1, \ldots, C_n)$ be an MCS and $M' = (C'_1, \ldots, C'_m)$ the mMCS where each C'_i is the management version of C_i. Then S is an equilibrium of M iff S is an equilibrium of M'.*

Example 6 (ctd). *A bacteria DL ontology C_3 and a generalized logic program C_4 to derive further possible drug effects are added, using suitable logics and semantics (belief sets for C_3 contain standard TBox subsumption inferences $subs(kb)$). Their knowledge bases are*

$$kb_3 = \{str\_pneu \sqsubseteq bact, leg\_pneu \sqsubseteq bact\},$$
$$kb_4 = \{eff(X, B) \text{:-}\ eff(X, A), isa(A, C), isa(B, C).\}.$$

As C_2 and C_3 have no bridge rules, they are simply cast into their management versions. C_4 gets bridge rules

$$\begin{aligned} br_4 = \{&add(isa(X, Y)) \leftarrow (3 : (X \sqsubseteq Y))\,.\\ &add(eff(X, B)) \leftarrow (1 : eff\ (X, B))\,.\\ &upd(not\ eff(X, B)) \leftarrow (1 : ineff\ (X, B))\,.\},\end{aligned}$$

and the bridge rules of C_1 are adapted: their heads use now the insert operation, and further rules that incorporate probable drug effects from C_4 are added:

$$\begin{aligned} br_1 = \{&insert(treat(X, B, I, likely)) \leftarrow (1{:}\ eff\ (X, B))\,,\\ &\qquad\qquad\qquad\qquad (2{:}\ B\ rdf{:}causes\ I)\,.\\ &insert(eff(X, B)) \leftarrow (4 : eff\ (X, B))\,.\}.\end{aligned}$$

The resulting mMCS $M_{ph} = (C_1, C_2, C_3, C_4)$ has one equilibrium $S = (S_1, S_2, S_3, S_4)$ where, omitting for brevity atoms of the form not a in S_4, $S_2 = inf_2(kb_2) = kb_2$, $S_3 = subs(kb_3)$,

$$\begin{aligned} S_1 = \{&treat(pen, str\_pneu, men, likely),\\ &treat(azith, leg\_pneu, atyp\_pneu, likely),\\ &treat(azith, str\_pneu, pneu, est),\\ &treat(azith, str\_pneu, men, est),\\ &eff(azith, str\_pneu), eff(azith, leg\_pneu),\\ &eff(pen, str\_pneu)\} \cup kb_1, \text{ and}\end{aligned}$$

$$\begin{aligned} S_4 = \{&eff(pen, str\_pneu), eff(azith, str\_pneu),\\ &eff(azith, leg\_pneu), isa(str\_pneu, bact),\\ &isa(leg\_pneu, bact)\}.\end{aligned}$$

4 Sample Instantiations

We consider instantiations of our framework, first discussing relational databases, logic programs, and belief revision. Second, we capture argumentation context systems by mMCSs.

Relational Databases. For relational databases, our running example already shows how a management function is used to realize view-updates. Many other operations on databases may be realized using managed contexts. In fact, e.g., the SQL language, Σ_{SQL}, can be accomodated: a context whose management base is built upon Σ_{SQL} and a management function mng_{SQL} which realizes the SQL semantics. This allows to use SQL in an mMCS. Observe that the implementation of mng_{SQL} is rather trivial, as existing implementations of SQL can be used via suitable interfaces, e.g., MySQL, Oracle DB, etc. In our running example, the respective view statement is:

```
CREATE VIEW eff AS
       SELECT drug, bacteria FROM treat
       WHERE credibility = evd;
```

To realize ordered sequences of SQL statements one may use timestamps in bridge rules handled by mng_{SQL}.

Belief Revision. Change of logical theories and knowledge bases is a long-standing area in logic and AI. Central operations on beliefs are *expansion*, *contraction*, *revision* and *update* (see Peppas [2008] for an excellent survey).

Let L be a logic with set $\mathcal{L}$ of formulas and semantics ACC_L, and let $rev : 2^{\mathcal{L}} \times \mathcal{L} \rightarrow 2^{\mathcal{L}}$ be a revision operator for theories in L. We may define a management function mng_{rev} for the management base $\{revise\}$, e.g., as follows: $mng_{rev}(O, kb) = \{(rev(kb, \{\phi_1 \wedge \ldots \wedge \phi_n \mid revise(\phi_i) \in O\}), ACC_L)\}$. Here, multiple revisions ($n \geq 2$) are handled by conjunction; other realizations (e.g., iteration) exist.

Logic Programming. Various extensions of logic programs have been proposed, e.g., updates of logic programs [Alferes *et al.*, 2002] which we use in our running example, debugging-support [Brain *et al.*, 2007], or meta-reasoning support. Many of them are realized using meta-programming, i.e., they transform a logic program P_e and additional input I, into a logic program P_t, such that solutions to the problem given by P_e and I are obtained from P_t (without altering the semantics). In the mMCS framework, we can achieve this directly using a management function mng such that, for a program P_e and operational statements O encoding I, $mng(O, P_e) = \{(P_t, ACC_{LP})\}$ where P_t is assembled from P_e and O, and ACC_{LP} is the employed semantics of logic programs.

Example 7 (ctd). *Suppose C_4 uses the update semantics of Alferes* et al. *[2002], i.e., the semantics of upd is given by the respective program transformation. For the operational statement $upd(not\ eff(pen, leg\_pneu))$ which is applicable in the belief state S of Example 6, the relevant rules are*

$$not\ eff(pen, leg\_pneu) : -\ . \tag{2}$$

$$eff(pen, leg\_pneu) : -\ eff(pen, bact), isa(str\_pneu, bact), isa(leg\_pneu, bact). \tag{3}$$

The ground instance (3) *is rejected (not contained in the transformed program), intuitively because it is in conflict with the more recent information represented by* (2)*. Therefore, the stable model does not contain $eff(pen, leg\_pneu)$.*

The ability of the management function to choose among different semantics allows one to flexibly select a suitable logic program semantics depending on the belief state. E.g., one may enforce for a program in context C_i that paraconsistent semantics as in Sakama and Inoue [1995] is used if it has an inconsistent belief set, and answer set semantics otherwise. It is achieved by self-referential bridge rules $mode\_pas(b) \leftarrow (i:\bot)$ and $mode\_as(b) \leftarrow not(i:\bot)$, where $\bot$ encodes inconsistency and $mode\_x$ the semantic mode of use.

Argumentation Context Systems. Argumentation context systems (ACS) [Brewka and Eiter, 2009] are homogeneous: all reasoning components are Dung argumentation frameworks (AFs) [Dung, 1995]. Yet, in this restricted setting they provide some functionality which is relevant here: bridge rules not only extend context AFs, they may also invalidate arguments or attack relations, they may even select a semantics; moreover, they provide ways of resolving inconsistent updates.

This additional functionality is achieved by introducing for each context a *mediator* $Med = (br, inc)$, which consists of a set br of bridge rules plus an inconsistency handling method inc. The heads of bridge rules are expressions in a genuine update description language. Based on inc, Med collects the heads of applicable bridge rules and returns one or several consistent sets of update statements, called acceptable updates.

A state S assigns to each context an update U_i and a set of arguments As_i. Equilibria (here called acceptable states) satisfy the following condition: each As_i must be an acceptable set of arguments for the context's AF updated with U_i, and U_i must be acceptable in S. It is not difficult to reconstruct this approach in the framework of mMCS. For each context C_i we choose O_i such that expressions of the ACS update language become operational statements. Furthermore, the management function mng_i must capture the meaning of the update expressions and the mediator's consistency handling method. Due to space limitations we cannot provide more detail but want to point out that acceptable ACS states and equilibria of the respective mMCS are in 1-to-1 correspondence.

5 Inconsistency Management

Different forms of inconsistency can arise in mMCS:

1. *Nonexistence of equilibria*: this global form of inconsistency was investigated in Eiter *et al.* [2010] for MCS.
2. *Local inconsistency:* even if equilibria exist, they may contain inconsistent belief sets. This presupposes an adequate notion of consistency (for belief sets and sets of formulas). In most context logics such a notion exists or is easily defined.
3. *Operator inconsistency:* the operations in the heads of applicable bridge rules are conflicting, e.g., operations like $add(p)$ and $delete(p)$, or $add(p)$ and $add(\neg p)$.

Handling inconsistencies of type 2 and 3 is one of the motivations that led to the development of mMCS. For type 1 inconsistencies the techniques from Eiter *et al.* [2010] can be adapted and, given suitable context managers, yield more adequate and precise means for diagnosis.

5.1 Local consistency

We now demonstrate how local consistency can be achieved by using adequate managers. We call a management function *mng local consistency (lc-) preserving* iff, for each set

O of operational statements and each KB kb, in every pair $(kb', ACC) \in mng(O, kb)$ the KB kb' is consistent. Furthermore, an mMCS M is *locally consistent* iff in each equilibrium $S = (S_1, \ldots, S_n)$ of M, all S_i are consistent belief sets.

Proposition 2. *Let M be an mMCS such that all management functions are lc-preserving. Then M is locally consistent.*

How to define lc-preserving managers? To simplify matters we assume all contexts are based on propositional logic with classical consistency and semantics, given by ACC_{pl}, and consider a single operator add with the obvious meaning. We proceed by: (1) selecting a base revision operator rev satisfying consistency preservation (revising a propositional KB with a consistent formula always results in a consistent KB); (2) picking maximal consistent subsets of the formulas to be added. Let $F_O = \{p \mid add(p) \in O\}$ and let $MC(F_O)$ be the set of maximal consistent subsets of F_O. Now define

$$mng(O, kb) = \{(rev\,(kb, \bigwedge F), ACC_{pl}) \mid F \in MC(F_O)\}\,.$$

This management function is obviously lc-preserving. Further refinements, e.g., based on additional preferences among bridge rules, are straightforward.

5.2 Global consistency

Previous work on inconsistency management in MCS (cf. Eiter *et al.* [2010]) introduced two basic notions: diagnoses and explanations. The former is consistency-based and intuitively characterizes bridge rules and corresponding modifications to obtain a consistent MCS, the latter is entailment-based and gives bridge rules that create resp. inhibit inconsistency. Both notions extend to mMCS.

Let br_M denote the set of all bridge rules of an mMCS M, let $hd(R) = \{hd(r) \mid r \in R\}$ for $R \subseteq br_M$, and let $M[R]$ denote the mMCS where only bridge rules in R occur.

A diagnosis is a pair of sets of bridge rules, such that: if the rules in the first set are deactivated, and the rules in the second set are added in unconditional form (an extremal form of strengthening sufficient to detect culprit rules) the mMCS becomes consistent (i.e., admits an equilibrium). Formally, given an mMCS M, a *diagnosis* of M is a pair (D_1, D_2), $D_1, D_2 \subseteq br_M$, s.t. $M[br_M \setminus D_1 \cup hd(D_2)] \not\models \bot$. $D^{\pm}(M)$ is the set of all such diagnoses and $D^{\pm}_m(M)$ denotes the set of all pointwise subset-minimal diagnoses.

An *explanation* is a pair of sets of bridge rules, such that their presence, resp. non-applicability, causes a relevant inconsistency in the given mMCS. Formally, given an mMCS M, an *explanation* of M is a pair (E_1, E_2) of sets $E_1, E_2 \subseteq br_M$ s.t. for all (R_1, R_2) where $E_1 \subseteq R_1 \subseteq br_M$ and $R_2 \subseteq br_M \setminus E_2$, it holds that $M[R_1 \cup hd(R_2)] \models \bot$. $E^{\pm}(M)$ denotes the set of all inconsistency explanations of M, and $E^{\pm}_m(M)$ the set of all pointwise subset-minimal ones.

Example 8 (ctd). *Suppose to improve M_{ph} by adding a further bridge rule $r_1 : insert(t) \leftarrow not(4 : t)$ to br_1, where $t = ineff(water,\ leg\_pneu)$. It should ensure that water is considered ineffective, even if C_4 does not derive this. The resulting mMCS M'_{ph} is inconsistent: if $t \notin S_4$, then $t \in S_1$ must hold, implying $t \in S_4$, a contradiction; if $t \in S_4$ then $t \notin S_1$ follows, which implies $t \notin S_4$, again a contradiction.*

Let r_2 be the rule of form $upd(t) \leftarrow (1 : t)$ in br_4. Then M'_{ph} has one minimal explanation, $(\{r_1, r_2\}, \{r_1, r_2\})$, and 4 minimal diagnoses: $(\{r_1\}, \emptyset)$, $(\{r_2\}, \emptyset)$, $(\emptyset, \{r_1\})$, $(\emptyset, \{r_2\})$.

One of the basic functions of context managers is to ensure an acceptable belief set of a context regardless of its applicable operational statements. We call a context C_i with knowledge base kb_i in an mMCs M *totally coherent* iff for every belief state S of M some $(kb', ACC_i) \in mng_i(app_i(S), kb_i)$ exists s.t. $ACC_i(kb') \neq \emptyset$; and C_i *totally incoherent* iff no belief state S fulfills the previous condition.

Note that any context with an lc-preserving management function is totally coherent; the opposite need not be the case.

Example 9 (ctd). *All contexts of M_{ph} and M'_{ph} are totally coherent. Indeed, for C_2 and C_3 this holds trivially, as they have no bridge rules. Similarly, for C_1 each insertion of tuples yields a knowledge base with an acceptable belief set (as eff and ineff may share tuples, the belief set may be regarded to be inconsistent). For C_4 we observe that the logic program has no stable model only if eff(X, B) and not eff(X, B) are derived for some (X, B). The atom not eff(X, B) however, can only hold for atoms added through the upd operation whose semantics guarantees that eff(X, B) is no longer derivable. Therefore C_4 has always an acceptable belief set.*

This shows that total coherence can not prevent inconsistency of the whole mMCS caused by cyclic information flow.

On the other hand, context managers can ensure the existence of diagnoses. To guarantee the existence of a diagnosis for an MCS M, Eiter *et al.* [2010] require that $M[\emptyset]$ is consistent. For mMCS we can replace this premise by the considerably weaker assumption that no context is totally incoherent.

Proposition 3. *Let M be an inconsistent mMCS. Then $D^{\pm}(M) \neq \emptyset$ iff no context of M is totally incoherent.*

The following proposition establishes that all possible inconsistencies are caused by cycles given that all context managers are totally coherent. Note, however, that not every cycle causes inconsistency, and that due to potential non-monotonicity inside contexts the number of negative beliefs in a cycle is not relevant. Let $M = (C_1, \ldots, C_n)$ be an mMCS, we write $ref_r(i, j)$ iff r is a bridge rule of context C_i and $(C_j : p)$ occurs in the body of r. For an mMCS M and $r_1, \ldots, r_k \in br_M$, we say that $(r_1, \ldots, r_k)$ forms a cycle iff $ref_{r_1}(i_1, i_2), \ldots, ref_{r_{k-1}}(i_{k-1}, i_k)$, and $ref_{r_k}(i_k, i_1)$ hold.

Proposition 4. *Let M be an inconsistent mMCS with totally coherent contexts. Then for every minimal explanation $(E_1, E_2) \in E^{\pm}_m(M)$ there exists a cycle $(r_1, \ldots, r_k)$ such that $E_1 = \{r_1, \ldots, r_k\}$.*

The (indirect) proof hinges on the following observations: if no cycle of rules in E_1 causes inconsistency, then there exists a 'leave' context C causing inconsistency, contradicting total coherence; by minimality the cycle contains all $r \in E_1$.

Example 10 (ctd). *For M'_{ph} and its minimal explanations $E^{\pm}_m(M'_{ph}) = \{(\{r_1, r_2\}, \{r_1, r_2\})\}$, (r_1, r_2) is a cycle.*

Importantly, for acyclic—in particular for hierarchic—mMCS, total coherency is sufficient for global consistency.

Corollary 1. *Any acyclic, mMCS with totally coherent contexts has an equilibrium.*

6 Complexity

We consider here the consistency problem $\mathcal{CONS}(M)$, i.e., given an mMCS $M = (C_1, \ldots, C_n)$, decide whether it has some equilibrium. For this, we can utilize *output-projected* belief states similar as in Eiter *et al.* [2010]. For context C_i, let OUT_i be the set of all beliefs p occurring in the body of some bridge rule in M. Then, the *output-projection* of a belief state $S = (S_1, \ldots, S_n)$ of M is the belief state $S' = (S'_1, \ldots, S'_n)$, $S'_i = S_i \cap OUT_i$, for $1 \leq i \leq n$. For consistency checking, we can concentrate on output-projections of equilibria:

Proposition 5. *An mMCS $M = (C_1, \ldots, C_n)$ is consistent iff some output-projected belief state $S' = (S'_1, \ldots, S'_n)$, exists such that, for all $1 \leq i \leq n$, $S'_i \in \{S_i \cap OUT_i \mid S_i \in ACC_i(kb'_i) \wedge (kb'_i, ACC_i) \in mng_i(app_i(S'), kb_i)\}$.*

Generalizing Eiter *et al.* [2010], let the *context complexity* of C_i be the complexity of the following problem:

(CC) Decide, given a set O_i of operator statements and $S'_i \subseteq OUT_i$, whether some $(kb'_i, ACC_i) \in mng_i(O_i, kb_i)$ and $S_i \in ACC_i(kb'_i)$ exist s.t. $S'_i = S_i \cap OUT_i$.

Here, C_i is explicitly represented by kb_i and br_i, and the logic suite is implicit, i.e., an oracle decides existence of S_i. The *context complexity* $\mathcal{CC}(M)$ of an mMCS M is a (smallest) upper bound for the context complexity classes of all C_i.

Depending on $\mathcal{CC}(M)$, the complexity of consistency checking for some complexity classes is shown in the following table, where $i \geq 1$, and entries denote membership results, resp. completeness results if CC is hard for some C_i:

$\mathcal{CC}(M)$ in	$\mathbf{P}$	$\mathbf{\Sigma^P_i}$	$\mathbf{\Delta^P_{i+1}}$	**PSPACE**	**EXPTIME**
$\mathcal{CONS}(M)$	**NP**	$\mathbf{\Sigma^P_i}$	$\mathbf{\Sigma^P_{i+1}}$	**PSPACE**	**EXPTIME**

Using simple insert/delete management, an example of $\mathcal{CC}(M)$ in $\mathbf{P}$ would be an mMCS built on defeasible logic (cf. Maher [2001]), and one for $\mathbf{NP}$ (resp., $\mathbf{\Sigma^P_2}$) using normal (disjunctive) answer set programs. Argumentation context systems [Brewka and Eiter, 2009] provide examples of mMCSs with context complexity in $\mathbf{\Delta^P_3}$; examples for $\mathbf{PSPACE}$ and $\mathbf{EXPTIME}$ can be found, e.g., among modal and description logics. Such contexts also have respective hard instances.

Problem CC intuitively consists of two subproblems: (MC) compute some $(kb'_i, ACC_i) \in mng_i(O_i, kb_i)$ and (EC) decide whether $S_i \in ACC_i(kb'_i)$ exists s.t. $S'_i = S_i \cap OUT_i$. However, it makes sense to analyze consistency depending on CC: often MC is solvable in polynomial time (perhaps nondeterministically and/or with the help of an oracle) or polynomial space, but kb'_i may become exponentially large (e.g., using a KB update or revision operator), nevertheless its explicit construction is avoidable for solving EC. If the output of MC remains polynomial, then the complexity of CC can suitably be characterized in terms of MC (e.g., maximal consistent subsets of a propositional theory) and EC. We leave more detailed results for an extended version.

7 Conclusion

Zhao *et al.* [2009] point out that simplified MCSs, where all bridge rules are of the form $a \leftarrow (c_i : a)$, are important for practical matters. Such MCSs are less expressive, as complex rule bodies and negation (thus nonmonotonic information flow) are not supported. In the presence of context managers, however, an analogous restriction does not impair expressivity: using, e.g., bridge rules $o_{(c_i,a)}(b) \leftarrow (c_i : a)$ with designated operations $o_{(c_i,a)}$, the management functions can emulate any bridge rule (1). In fact, every MCS M can be easily transformed into such an mMCS M' having the same equilibria.

An interesting issue for future work are refined semantics for mMCS to discriminate among equilibria, such as an extension of minimal or grounded equilibria [Brewka and Eiter, 2007] to mMCS. Moreover, as mMCS may yield alternative knowledge bases, preference of equilibria may be based on preference of alternatives. In particular for consistency restoring, minimality of change seems natural. Investigating this and developing mMCS implementations are on our agenda.

References

[Alferes *et al.*, 2002] J.-J. Alferes, L.-M. Pereira, H. Przymusinska, and T.C. Przymusinski. LUPS—A language for updating logic programs. *AIJ*, 138(1-2):87–116, 2002.

[Bikakis and Antoniou, 2008] A. Bikakis, and G. Antoniou. Distributed Defeasible Contextual Reasoning. In *Proc. AmI*, pp 308–325, 2008.

[Brain *et al.*, 2007] M. Brain, M. Gebser, J. Pührer, T. Schaub, H. Tompits, and S. Woltran. Debugging ASP programs by means of ASP. In *Proc. LPNMR*, pp 31–43, 2007.

[Brewka and Eiter, 2007] G. Brewka and T. Eiter. Equilibria in heterogeneous nonmonotonic multi-context systems. In *Proc. AAAI*, pp 385–390, 2007.

[Brewka and Eiter, 2009] G. Brewka and T. Eiter. Argumentation context systems: A framework for abstract group argumentation. In *Proc. LPNMR*, pp 44–57, 2009.

[Dung, 1995] P.M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *AIJ*, 77:321–358, 1995.

[Eiter *et al.*, 2010] T. Eiter, M. Fink, P. Schüller, and A. Weinzierl. Finding explanations of inconsistency in multi-context systems. In *Proc. KR*, pp 329–339, 2010.

[Giunchiglia and Serafini, 1994] F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics or: How we can do without modal logics. *AIJ*, 65(1):29–70, 1994.

[Maher, 2001] M.J. Maher. Propositional defeasible logic has linear complexity. *TPLP*, 1(6):691–711, 2001.

[Peppas, 2008] P. Peppas. Belief revision. In van Harmelen et al. [2008], ch. 8, pp 317–360.

[Sakama and Inoue, 1995] C. Sakama and K. Inoue. Paraconsistent stable semantics for extended disjunctive programs. *J. Log. Comput.*, 5(3):265–285, 1995.

[van Harmelen *et al.*, 2008] F. van Harmelen, V. Lifschitz, and B. Porter, editors. *Handbook of Knowledge Representation*. Elsevier, 2008.

[Zhao *et al.*, 2009] Y. Zhao, K. Wang, R. Topor, J.Z. Pan, and F. Giunchiglia. Building heterogeneous multi-context systems by semantic bindings. Tech. rep., Ingegneria e Scienza dell'Informazione, University of Trento, 2009.