The MCS-IE System for Explaining Inconsistency in Multi-Context Systems*

Markus Bögl, Thomas Eiter, Michael Fink, and Peter Schüller

Institute of Information Systems
Vienna University of Technology
Favoritenstrasse 11, A-1040 Vienna, Austria
markus.boegl@student.tuwien.ac.at, {eiter,fink,schueller}@kr.tuwien.ac.at

Abstract. The Multi-Context System Inconsistency Explainer allows for evaluation of semantics and explanation of inconsistencies in systems where heterogeneous knowledge bases are linked via nonmonotonic rules. The implementation is based on the dlvhex tool, which is an extension of answer set programming with external atoms and higher order features.

1 Introduction

Nonmonotonic multi-context systems (MCSs) were introduced in [1], mainly but not exclusively as an extension of [5, 2]. They are a formalism for interlinking heterogeneous knowledge bases (called contexts), whose semantics is defined via (possibly non-unique) belief sets, via bridge rules that may be non-monotonic. For example, $(2 : b) \leftarrow (1 : a), not(3 : a)$ expresses that context 2 should add b to its knowledge base, if context 1 has a in its belief set while context 3 has not. The semantics of MCSs is then defined in terms of belief states, which contain one belief set per context, that satisfy a stability condition (called equilibria).

Inconsistency is the absence of such equilibria. An inconsistent MCS yields no information, therefore our aim is to explain reasons for inconsistency in MCSs in order to support users in dealing with inconsistency. For this purpose, the notions of *diagnosis* and *inconsistency explanation* were introduced in [3].

A diagnosis (D_1, D_2) points out a set of bridge rules D_1 which must be removed from an MCS M, and a set of bridge rules D_2 whose unconditional form (i.e., $\alpha \leftarrow$ for $\alpha \leftarrow \beta$) must be added to M to restore consistency in M. An explanation (E_1, E_2) points out a set of bridge rules E_1 which is required in M, and a set of bridge rules E_2 whose unconditional forms must not be added to M to ensure inconsistency in M. Diagnoses allow to find overall system repairs, whereas inconsistency explanations separate individual sources of inconsistency.

A method for evaluating these notions by a rewriting to logic programming was introduced in [3]. This work describes the MCS Inconsistency Explainer system MCS-IE, which realizes and extends the implementation concepts of [3].[1]

* This work was supported by the Vienna Science and Technology Fund (WWTF) under grant ICT08-020.

[1] http://www.kr.tuwien.ac.at/research/systems/mcsie/

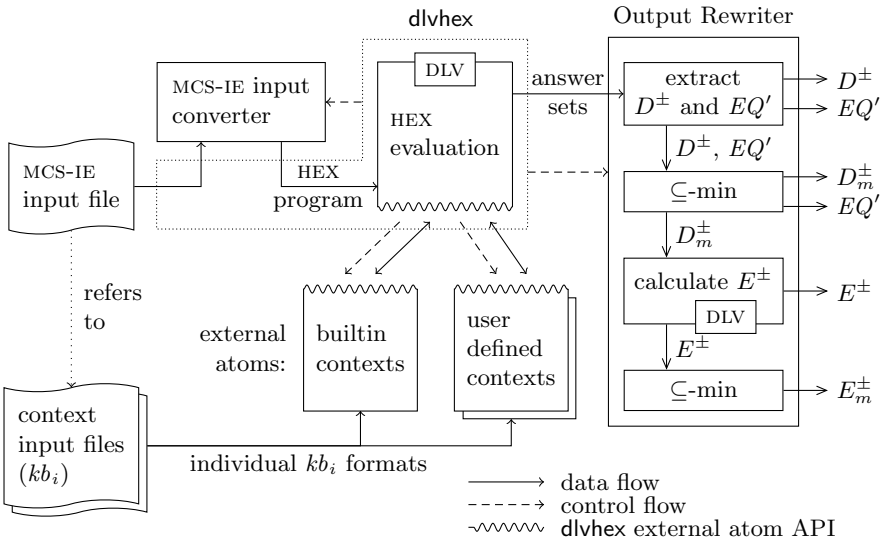

Fig. 1. Data flow between MCS-IE components, dlvhex and DLV.

Our system uses the dlvhex solver[2] and HEX programs, which are an extension of answer set programs (ASPs) with external atoms [4]. We use external atoms for capturing context semantics, since each context of an MCS can be based on its own logic, and in general cannot be rewritten to ASPs.

The MCS-IE system provides the following features:

- computation of MCS semantics (output projected equilibria),
- computation of diagnoses, explanations, and their subset-minimal notions,
- support for contexts formalized in ASP (specifically: DLV programs), and
- an API in C++ for integrating user-defined contexts.

2 System Architecture

Figure 1 shows the architecture of the MCS-IE system. It is implemented as a plugin to dlvhex, therefore dlvhex controls all MCS-IE components.

To analyze inconsistency in an MCS, a master input file which describes the MCS topology (bridge rules and contexts) must be provided by the user. dlvhex rewrites the master file into a HEX program using the MCS-IE input converter. For space reasons, we refer to the MCS-IE website for details of this rewriting.[1] The main idea is to guess a diagnosis using auxiliary predicates, to guess a belief state, and to evaluate bridge rule semantics based on the guesses. Context semantics are then evaluated using dlvhex external atoms, which may require additional context input files (e.g., ASP program files, or databases).

[2] `http://www.kr.tuwien.ac.at/research/systems/dlvhex/`

Each answer set of the rewritten HEX program describes a diagnosis of the given MCS and a corresponding equilibrium. The MCS-IE output rewriter component extracts this information and converts it into a human readable format (see the example below). For the conversion of diagnoses to inconsistency explanations the output converter generates an answer set program from the extracted diagnosis, and evaluates it using DLV. Depending on command-line parameters,[1] sets of (subset-minimal) explanations for inconsistency in the MCS are displayed.

3 Example

In the following, we give an example MCS along with its encoding for MCS-IE.

Our MCS is a health care decision support system, which consists of a database of lab test results C_1, a patient record database C_2, an ontology C_3 for disease classification, and an expert system C_4 suggesting suitable treatments.

The MCS topology is specified in file `master.hex`, contexts C_1, C_2, and C_4 are realized in corresponding DLV program files:

master.hex:
```
#context(1,"dlv_asp_context_acc", "kb1.dlv").
#context(2,"dlv_asp_context_acc", "kb2.dlv").
#context(3,"ontology_context3_acc", "").
#context(4,"dlv_asp_context_acc", "kb4.dlv").
r1: (3:pneum)           :- (2:xraypneum).
r2: (3:marker)          :- (2:marker).
r3: (4:need_ab)         :- (3:pneum).
r4: (4:need_strong)     :- (3:atyppneum).
r5: (4:allow_strong_ab) :- not (1:allergystrong).
```

kb1.dlv:
```
allergystrong.
```

kb2.dlv:
```
marker. xraypneum.
```

kb4.dlv:
```
give_strong v give_weak :- need_ab.
give_strong             :- need_strong.
give_nothing            :- not need_ab, not need_strong.
:- give_strong, not allow_strong_ab.
```

Intuitively, C_1 and C_2 provide information that the patient is allergic to strong antibiotics, that a certain blood marker is present, and that pneumonia was detected in an X-ray examination. C_4 suggests a treatment which is either a strong antibiotic, a weak antibiotic, or no medication at all.

Context C_3 is implemented in C++ and uses the MCS-IE API. The following source code (namespaces and includes are omitted) builds into a `dlvhex` plugin:

```
DLVHEX_MCSEQUILIBRIUM_PLUGIN(MedExamplePluginContext3,0,1,0)
DLVHEX_MCSEQUILIBRIUM_CONTEXT(Context3,"ontology_context3_acc")
set<set<string> > Context3::acc(
      const string& param, const set<string>& input) {
  set<set<string> > ret;
  set<string> s(input.begin(), input.end());
  if( input.count("pneum") == 1 && input.count("marker") == 1 )
```

```
    s.insert("atyppneum");
  ret.insert(s); return ret;
}
void MedExamplePluginContext3::registerAtoms()
  { registerAtom<Context3>(); }
```

The first two lines creates a `dlvhex` plugin and an external atom usable in `#context(...)`. Context semantics is implemented in function `acc`, which gets bridge rule heads as `input` and returns accepted belief sets. Finally, the external atom is registered in the plugin using `registerAtoms`.

Roughly, C_3 specifies that presence of pneumonia together with a blood marker (stemming from r_2) yields atypical pneumonia in the belief state.

Note that this system is inconsistent, as the expert system concludes that a patient must be given a special drug, but the patient record states that she is allergic to that drug, a counter-indication. This inconsistency can be explained using MCS-IE as follows (assuming the MCS-IE plugin in the current directory):

```
$ dlvhex --plugindir=./ --ieenable --ieexplain=Dm,Em master.hex
```

MCS-IE calculates the following output, containing minimal diagnoses plus witnessing equilibria (`Dm:EQ:`), and minimal inconsistency explanations (`Em`):

```
Dm:EQ:({r1},{}):({allergystrong},{marker,xraypneum},{})
Dm:EQ:({r2},{}):({allergystrong},{marker,xraypneum},{pneum},{})
Dm:EQ:({r4},{}):({allergystrong},{marker,xraypneum},{atyppneum,pneum},{})
Dm:EQ:({},{r5}):({allergystrong},{marker,xraypneum},{atyppneum,pneum},{})
Em:({r1,r2,r4},{r5})
```

Accordingly, deactivating r_1, or r_2, or r_4, or adding r_5 unconditionally, will restore consistency, and there is a single inconsistency involving all rules except for r_3.

4 Conclusions

The MCS-IE system may serve as a valuable tool for analyzing inconsistencies in MCSs. It is geared towards functionality, not (yet) towards efficiency. An interactive demo is available.[1] Future work will be the implementation of further external atoms, e.g., for accessing DBMS.

References

1. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: AAAI. pp. 385–390 (2007)
2. Brewka, G., Roelofsen, F., Serafini, L.: Contextual default reasoning. In: IJCAI. pp. 268–273 (2007)
3. Eiter, T., Fink, M., Schüller, P., Weinzierl, A.: Finding explanations of inconsistency in nonmonotonic multi-context systems. In: KR. pp. 329–339 (2010)
4. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In: IJCAI. pp. 90–96 (2005)
5. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics, or: How we can do without modal logics. Artificial Intelligence 65(1), 29–70 (1994)