The DMCS Solver for Distributed Nonmonotonic Multi-Context Systems*

Seif El-Din Bairakdar, Minh Dao-Tran, Thomas Eiter, Michael Fink, and Thomas Krennwallner

Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
{bairakdar,dao,eiter,fink,tkren}@kr.tuwien.ac.at

1 Introduction

The DMCS system is an implementation of the equilibrium semantics for heterogeneous and nonmonotonic multi-context systems (MCS) [3], which feature contexts with heterogeneous and possibly nonmonotonic logics. Each context in an MCS comprises of two parts: a local knowledge base and a set of bridge rules that can access the beliefs of other contexts and add new information to the knowledge base. In this setting, contexts are loosely coupled, and may model distributed information linkage applications; thus it is natural to have a system that allows for the distributed evaluation of MCS.

In an MCS $M = (C_1, \ldots, C_n)$, each context C_i is characterized by a knowledge base kb_i and a set of bridge rules br_i. In our implementation, each kb_i is in DLV syntax as in [6]. The br_i are sets of nonmonotonic rules

$$p_0 \leftarrow (c_1 : p_1), \ldots, (c_j : p_j), \mathbf{not}\ (c_{j+1} : p_{j+1}), \ldots, \mathbf{not}\ (c_m : p_m).$$

where the $(c_k : p_k)$ are bridge atoms; the index c_k refers to a context C_{c_k} and p_k is a possible belief of C_{c_k}; intuitively, the atom is true if p_k is in the belief set of context C_{c_k}. If the body evaluates to true with respect to a belief state, which is a sequence $S = (S_1, \ldots, S_n)$ of belief sets S_i of C_i, $1 \leq i \leq n$, then p_0 has to be added to kb_i. The semantics of M is then given in terms of stable belief sets (called equilibria). *Partial Equilibria* are equilibria in a sub-MCS of M induced by a single context C_k resp. a collection $C_{k_1}, \ldots, C_{k_j}$ of contexts.

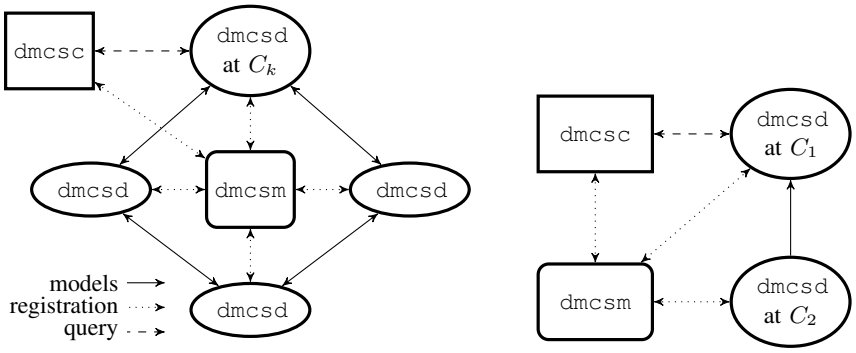
Example 1. Consider an MCS $M = (C_1, C_2)$, where C_1, C_2 have answer set programs in the local knowledge bases; specifically

$kb_1 = \{a_1 \leftarrow b_1; \bot \leftarrow \text{not } b_1\}$ and $br_1 = \{b_1 \leftarrow (2 : a_2)\}$;

$kb_2 = \{a_2 \vee b_2 \leftarrow\}$ and $br_2 = \emptyset$.

Then $S = (\{a_1, b_1\}, \{a_2\})$ is the only equilibrium of M; as well as the only partial equilibrium of M w.r.t. C_1. Note that w.r.t. C_2, M has two partial equilibria: $S^{(1)} = (\epsilon, \{a_2\})$ and $S^{(2)} = (\epsilon, \{b_2\})$ (here ϵ means the context is not reachable).

* This research has been supported by the Austrian Science Fund (FWF) project P20841 and by the Vienna Science and Technology Fund (WWTF) project ICT 08-020.



(a) System Architecture (b) Components of Ex. 2

Fig. 1: DMCS System Architecture

The algorithm in [4] describes a generic distributed procedure for evaluating (partial) equilibria of multi-context systems. It has been refined with an effective decomposition technique in [1]. Our DMCS system comprises both algorithms and allows for MCS with contexts that have propositional answer set programs as knowledge bases. Initial experimental results with the DMCS system were shown in [1, 4].

The basic idea for our system [4] is to take the bridge rules and the knowledge base of a context, compile them to a propositional theory, and use a SAT solver to compute the models. The distributed algorithms then take care of combining the models and generate equilibria at the context that initiated the computations.

DMCS is a purely distributed framework written in C++. It uses *clasp* [5] for local model building. The system is available at

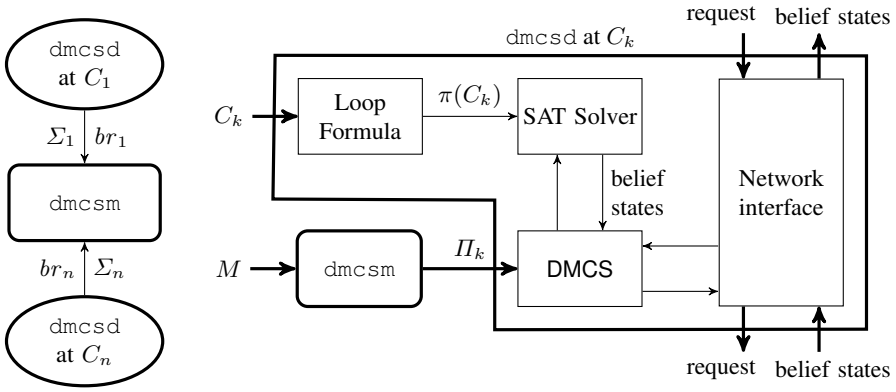
`http://www.kr.tuwien.ac.at/research/systems/dmcs/`.

2 System Architecture and Evaluation

The architecture of DMCS is outlined in Figure 1a, which has the following main components: (i) a front-end `dmcsc` for querying the multi-context system; (ii) daemons `dmcsd`, where each of them represents a context and interacts with the others; a daemon has four modules, namely Loop Formula, SAT Solver, DMCS, and Network Interface (cf. Figure 2b); and (iii) a component `dmcsm` holding meta information about the MCS that has been collected from each context. The system has four stages which are briefly described as follows:

System start-up. At this stage, all running `dmcsd` processes register at the `dmcsm`, provide their own set of bridge rules, alphabet as well as port and host name (Figure 2a). With this information, the `dmcsm` component identifies the topology of the system and gets ready to answer any question regarding this meta knowledge.

Initialization of `dmcsd`. Upon initialization, each `dmcsd` utilizes the Loop Formula module to transform its local knowledge base and bridge rules into a SAT theory denoted by $\pi(C_k)$ in DIMACS format (see [4] for details). Then, it starts listening for incoming requests from other daemons, or from queries of `dmcsc` described next.



(a) Start-up (b) dmcsd System Architecture

Fig. 2: DMCS System

Querying the system. When the user wants to know all partial equilibria of the system w.r.t. a starting context C_k, she uses dmcsc to pose the query. She may specify variables, which will be provided as the initial request to C_k. First, dmcsc inquiries the dmcsm component about C_k and gets back the connection settings of this context. Then, dmcsc sends the query to the respective dmcsd representing C_k and waits for the results.

Evaluating the System. After dmcsc has sent a query to the dmcsd process that represents the starting context C_k, the daemon computes partial belief states w.r.t. interface variables and projects unwanted variables away. If C_k needs beliefs from neighboring contexts, it sends a request to them and awaits their belief states, which will be consistently combined with the local beliefs of C_k. Essentially, those requests look just as queries sent from dmcsc, and every dmcsd will process them in a uniform manner. After all neighbors have been addressed, C_k returns the partial equilibria to the client, who presents them to the user.

The algorithm used in dmcsd is an ASP logic instance of the generic DMCS algorithm presented in [4]. Alternatively, dmcsd may use an adapted version of this algorithm, DMCSOPT, which exploits dependencies in the MCS, uses economically small representations of them, and uses minimal interface variables needed for minimizing data transmission (see [1]). Here, query plans Π_k w.r.t. C_k are key for guiding the evaluation process and may be provided by dmcsm.

3 System Usage

For a concrete usage scenario of DMCS, we reconsider the MCS in the example above.

Example 2 (cont'd). In order to evaluate our example, one has to set up a system as illustrated in Figure 1b, by executing two start up calls, possibly on different machines.

```
$ dmcsd --context=1 --kb=C1.kb --br=C1.br --manager=HOST:PORT
$ dmcsd --context=2 --kb=C2.kb --br=C2.br --manager=HOST:PORT
```

The command-line argument `--context` tells the daemon the context id that it will represent. The knowledge base and the bridge rule files are provided via `--kb` and `--br`, resp. The `--manager` option is used to set up the location of the `dmcsm` component.

To compute equilibria of M w.r.t. context C_1, the user queries the `dmcsm` to get the connection settings for the `dmcsd` representing C_1 using the following command (the parameters have the same meaning as above).

```
$ dmcsc --context=1 --manager=HOST:PORT
```

After `dmcsd` at C_1 finishes its computations, it delivers the result back to `dmcsc`. A list of the equilibria is then enumerated to the user:

```
( {a1,b1}, {a2} )
Total Number of Equilibria: 1
```

It is possible to specify for DMCS a set of atoms of interest; other atoms will then be discarded. Unless such a set is given, `dmcsc` will assume its default settings and proceed with standard operations.

4 Conclusions

The DMCS system is, to the best of out knowledge, the first implementation of a fully distributed algorithm to evaluate heterogeneous and nonmonotonic multi-context systems. Other related systems like the one in [7] does not allow cyclic references in bridge rules, and the system in [2] is based on a query evaluation approach.

The method for computing partial equilibria induced by some context can be easily extended to compute equilibria of the whole system; this, however, may be of less interest from the perspective of an individual context (e.g., in a peer-to-peer style evaluation).

Our ongoing work aims at further extending the implementation and optimization, as well as on dynamic configuration of MCS by instantiating generic bridge rules.

References

1. Bairakdar, S., Dao-Tran, M., Eiter, T., Fink, M., Krennwallner, T.: Decomposition of distributed nonmonotonic multi-context systems. In: JELIA'10. Springer (September 2010)
2. Bikakis, A., Antoniou, G., Hassapis, P.: Strategies for contextual reasoning with conflicts in ambient intelligence. Knowl Inf Syst. (2010), published online: 9 April 2010
3. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: AAAI'07. pp. 385–390. AAAI Press (July 2007)
4. Dao-Tran, M., Eiter, T., Fink, M., Krennwallner, T.: Distributed nonmonotonic multi-context systems. In: KR'10. pp. 60–70. AAAI Press (May 2010)
5. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-driven answer set solving. In: IJCAI'07. pp. 386–392. AAAI Press (January 2007)
6. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The dlv system for knowledge representation and reasoning. ACM Trans. Comput. Logic 7(3), 499–562 (2006)
7. Serafini, L., Tamilin, A.: Drago: Distributed reasoning architecture for the semantic web. In: ESWC'05. pp. 361–376. Springer (May 2005)