# Decomposition of 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

**Abstract.** Multi-Context Systems (MCS) are formalisms that enable the interlinkage of single knowledge bases, called contexts, via bridge rules. Recently, a fully distributed algorithm for evaluating heterogeneous, nonmonotonic MCS was described in [7]. In this paper, we continue this line of work and present a decomposition technique for MCS which analyzes the topology of an MCS. It applies pruning techniques to get economically small representations of context dependencies. Orthogonal to this, we characterize minimal interfaces for information exchange between contexts, such that data transmissions can be minimized. We then present a novel evaluation algorithm that operates on a query plan which is compiled with topology pruning and interface minimization. The effectiveness of the optimization techniques is demonstrated by a prototype implementation, which uses an off-the-shelf SAT solver and shows encouraging experimental results.

## 1 Introduction

In the last years, there has been increasing interest in systems comprising multiple knowledge bases. The rise of distributed systems and the World Wide Web fostered this development, and to date, several formalisms are available that accommodate multiple, possibly distributed knowledge bases. One formalism are Multi-Context Systems (MCS) consisting of several theories (the contexts) that are interlinked with bridge rules which allow to add knowledge to a context depending on knowledge in other contexts. E.g., the bridge rule $a \leftarrow (2:b)$ of a context $C_1$ means that $C_1$ should conclude $a$ if context $C_2$ believes $b$. MCS have applications in various areas, such as argumentation, data integration, or multi-agent systems. There, contexts may model the beliefs of an agent while the bridge rules model an agent's perception of the environment, i.e., other contexts.

Among the various MCS proposals (e.g., [10,11,12]), the general MCS framework of [5] is of special interest, as it generalizes previous approaches in contextual reasoning and allows for *heterogeneous and nonmonotonic* MCS, i.e., with different, possibly nonmonotonic logics in its contexts (thus furthering heterogeneity), and bridge rules

[⋆] 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.

may use default negation (to deal, e.g., with incomplete information). Hence, nonmonotonic MCS interlinking monotonic context logics are possible. This MCS framework can conveniently capture the following scenario, which we use as a running example.

*Example 1.* A group of four scientists, Ms. 1, Mr. 2, Mr. 3, and Ms. 4, just finished their conference visit and are now arranging a trip back home. They can choose between going by train or by car (which is usually slower than the train); and if they use the train, they should bring along some food. Moreover, Mr. 3 and Ms. 4 have additional information from home that might affect their decision.

Mr. 3 has a daughter, Ms. 6. He is fine with either transportation option, but if Ms. 6 is sick then he wants to use the fastest vehicle to get home. Ms. 4 just got married, and her husband, Mr. 5, wants her to come back as soon as possible. He urges her to try to come home even sooner, while Ms. 4 tries to yield to her husband's plea.

If they go by train, Mr. 3 is responsible for buying provisions. He might choose either salad or peanuts. The options for beverages are coke or juice. Mr. 2 is a modest person as long as he gets home. He agrees to any choice that Mr. 3 and Ms. 4 select for vehicle but he dislikes coke. Ms. 1 is the leader of the group and prefers to go by car, but if Mr. 2 and 3 go by train then she would not object. A problem is that Ms. 1 is allergic to nuts.

Mr. 3 and Ms. 4 do not want to bother the group with their circumstances and communicate just their preferences, which is sufficient for reaching an agreement. Ms. 1 decides which option to take based on the information she gets from Mr. 2 and Mr. 3.

Similar scenarios have already been investigated in the realm of multi-agent systems (see, e.g., [6] on social answer set programming). We do not aim at introducing a new semantics for such scenarios; our example is meant to be a plain showcase application of MCS. We stress that MCS have potential as a host for KR formalisms, just like answer set programs have; however, in this paper we concentrate on efficient MCS evaluation.

The distributed algorithm introduced in [7], called DMCS, computes the semantics of an MCS, which is given in terms of equilibria. Roughly, an equilibrium is a collection of local models (belief sets) for the individual contexts that is compatible with the bridge rules. The principle of the algorithm is, starting from context $C_k$ (the root), that models will be processed at each context. Bridge rules, which access beliefs in other contexts, implicitly span belief import dependencies between contexts. This relationship is used to navigate the system, and models returned from invoked neighbors are combined with the local beliefs and passed back to the invoking contexts. DMCS uses a parameter for projecting models to relevant variables to reduce data payload.

Experiments for an instantiation of DMCS with answer set programming contexts revealed some scalability issues which can be tracked down to the following problems:

(1) contexts are unaware of context dependencies in the system beyond their neighbors, and thus treat each neighbor in a generic way. Specifically, cyclic dependencies remain undetected until a context, seeing the invocation chain, requests models from a context in the chain. Furthermore, a context $C_k$ does not know whether a neighbor $C_i$ already requests models from another neighbor $C_j$ which then would be passed to $C_k$; hence, $C_k$ makes possibly a superfluous request to $C_j$.