Symmetry Breaking for Distributed Multi-Context Systems*

Christian Drescher[1], Thomas Eiter[2], Michael Fink[2],
Thomas Krennwallner[2], and Toby Walsh[1]

[1] NICTA and University of New South Wales
Locked Bag 6016, Sydney NSW 1466, Australia
{christian.drescher,toby.walsh}@nicta.com.au
[2] Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
{eiter,fink,tkren}@kr.tuwien.ac.at

Abstract. Heterogeneous nonmonotonic multi-context systems (MCS) permit different logics to be used in different contexts, and link them via bridge rules. We investigate the role of symmetry detection and symmetry breaking in such systems to eliminate symmetric parts of the search space and, thereby, simplify the evaluation process. We propose a distributed algorithm that takes a local stance, i.e., computes independently the partial symmetries of a context and, in order to construct potential symmetries of the whole, combines them with those partial symmetries returned by neighbouring contexts. We prove the correctness of our methods. We instantiate such symmetry detection and symmetry breaking in a multi-context system with contexts that use answer set programs, and demonstrate computational benefit on some recently proposed benchmarks.

1 Introduction

Due to the increasing application of distributed systems, there has been recent interest in formalisms that accommodate several, distributed knowledge bases. Based on work by McCarthy [14] and Giunchiglia [11], a powerful approach is multi-context systems (MCS; [12]). Intuitively, an MCS consists of several heterogeneous theories (the contexts), which may use different logical languages and different inference systems, that are interlinked with a special type of rules that allow to add knowledge into a context depending on knowledge in other contexts. MCSs have applications in various areas such as argumentation, data integration, and multi-agent systems. In the latter, each context models the beliefs of an agent while the bridge rules model an agent's perception of the environment. Among various proposals for MCS, the general MCS framework of Brewka and Eiter [5] is of special interest, as it generalises previous approaches in contextual reasoning and allows for heterogeneous and nonmonotonic MCSs. Such a system can have different, possibly nonmonotonic logics in the different contexts, e.g.,

* This research has been supported by the Austrian Science Fund project P20841 and by the Vienna Science and Technology Fund project ICT 08-020. NICTA is funded by the Department of Broadband, Communications and the Digital Economy, and the Australian Research Council.

answer set programs (ASP; [4]), and bridge rules can use default negation to deal with incomplete information.

Although there has been dramatic improvements [3] in the performance of distributed algorithms for evaluating Brewka and Eiter' style nonmonotonic MCSs such as DMCS [7], many applications exhibit symmetries. For example, suppose context C_1 is an advanced database system which repairs inconsistencies (e.g., from key violations in database tables), and another context C_2 is accessing the repaired tables via bridge rules. A large (exponential) number of repairs may exist, each yielding a local model (i.e., belief set) of C_1; many of those models are symmetric, thus C_2's bridge rules may fire for many symmetric repairs. This can frustrate an evaluation algorithm as it fruitlessly explores symmetric subspaces. Furthermore, communicating symmetric solutions from one context to another can impede further search. If symmetries can be identified, we can avoid redundant computation by pruning parts of the search space through symmetry breaking. However, symmetry breaking in MCSs has not been explored in any depth.

In order to deal with symmetry in MCSs, we must accomplish two tasks: (1) identifying symmetries and (2) breaking the identified symmetries. We make several fundamental and foundational contributions to the study of symmetry in MCS.

- First, we define the notion of symmetry for MCSs. This is subsequently specialized to local symmetries and partial symmetries that capture symmetry on parts of an MCS. Partial symmetries can be extended to a symmetry of the whole system under suitable conditions which are formalized in a corresponding notion of join.
- Second, we design a distributed algorithm to identify symmetries based on such partial symmetries. The method runs as background processes in the contexts and communicate with each other for exchanging partial symmetries. This algorithm computes symmetries of a general MCS based on the partial symmetries for each individual context. We demonstrate such symmetry detection for ASP contexts using automorphisms of a suitable coloured graph.
- Third, we break symmetries by extending the symmetry breaking methods of Crawford *et al.* [6] to distributed MCS. We construct symmetry-breaking constraints (SBCs) for a MCS that take into account beliefs imported from other contexts into account. These constraints ensure that an evaluation engine never visits two points in the search space that are symmetric. For contexts other than propositional logic, distributed SBCs have to be expressed appropriately. Again we illustrate this in the case of ASP contexts and develop a logic-program encoding for distributed symmetry breaking constraints.
- Finally, we experimentally evaluate our approach on MCSs with ASP contexts. In problems with large number of symmetries, we demonstrate the effectiveness of only breaking a subset of the symmetries. Results on MCS benchmarks that resemble context dependencies of realistic scenarios [3] show that symmetry breaking yields significant improvements in runtime and compression of the solution space.

2 Logical Background

We recall some basic notions of heterogeneous nonmonotonic multi-context systems. Following [5], a *logic* over an alphabet $\mathcal{A}$ is a triple $L = (\mathrm{KB}, \mathrm{BS}, \mathrm{ACC})$, where KB is

a set of well-formed knowledge bases over $\mathcal{A}$, BS is a set of possible belief sets (sets over $\mathcal{A}$), and ACC: KB $\to 2^{\text{BS}}$ is a function describing the semantics of the logic by assigning each $kb \in$ KB a set of acceptable sets of beliefs. This covers many monotonic and nonmonotonic logics like *propositional logic* under the closed world assumption and *default logic*. We concentrate on logic programs under answer set semantics, i.e., ASP logic L. A (disjunctive) *logic program* over an alphabet $\mathcal{A}$ is a finite set of rules

$$a_1; \ldots ; a_\ell \leftarrow b_1, \ldots, b_j, \sim b_{j+1}, \ldots, \sim b_m \tag{1}$$

where $a_i, b_k \in \mathcal{A}$ for $1 \leq i \leq \ell$, and $1 \leq k \leq m$. A *literal* is an atom a or its default negation $\sim a$. For a rule r, let head$(r) = \{a_1, \ldots, a_\ell\}$ be the *head* of r and body$(r) = \{b_1, \ldots, b_j, \sim b_{j+1}, \ldots, \sim b_m\}$ the *body* of r. For an ASP logic L, the set of knowledge bases KB is given through the set of logic programs, the possible belief sets BS $= 2^{\mathcal{A}}$ contains all subsets of atoms, and ACC(P) is the set of answer sets of a logic program P. For a detailed introduction to ASP, we refer to [4].

We now recall multi-context systems according to Brewka and Eiter [5]. A *multi-context system* $M = (C_1, \ldots, C_n)$ consists of a collection of contexts $C_i = (L_i, kb_i, br_i)$, where $L_i = (\text{KB}_i, \text{BS}_i, \text{ACC}_i)$ is a logic over alphabets $\mathcal{A}_i$, $kb_i \in \text{KB}_i$ is a knowledge base, and br_i is a set of L_i *bridge rules* r of the form

$$a \leftarrow (c_1 : b_1), \ldots, (c_j : b_j), \sim (c_{j+1} : b_{j+1}), \ldots, \sim (c_m : b_m) \ , \tag{2}$$

where $1 \leq c_k \leq n$, b_k is an atom in $\mathcal{A}_{c_k}$, $1 \leq k \leq m$, and $kb \cup \{a\} \in \text{KB}_i$ for each $kb \in \text{KB}_i$. We call a *context atom* $(c_k : b_k)$ or its default negation $\sim (c_k : b_k)$ a *context literal*. Analogous to standard notions of ASP, let the atom head$(r) = a$ be the *head* of r and body$(r) = \{(c_1 : b_1), \ldots, (c_j : b_j), \sim (c_{j+1} : b_{j+1}), \ldots, \sim (c_m : b_m)\}$ the *body* of r. For a set S of context literals, define $S^+ = \{(c : b) \mid (c : b) \in S\}$, $S^- = \{(c : b) \mid \sim (c : b) \in S\}$, and for a set S of context atoms, let $S|_c = \{b \mid (c : b) \in S\}$. The set of atoms occurring in a set br_i of bridge rules is denoted by at(br_i). W.l.o.g., we will assume that the alphabets $\mathcal{A}_i$ are pairwise disjoint and denote their union by $\mathcal{A} = \bigcup_{i=1}^{n} \mathcal{A}_i$.

Intuitively, context literals in bridge rules refer to information of other contexts. Bridge rules can thus modify the knowledge base, depending on what is believed or disbelieved in other contexts. The semantics of an MCS is given by its equilibria, which is a collection of acceptable belief sets, one from each context, that respect all bridge rules. More formally, for an MCS $M = (C_1, \ldots, C_n)$ define a *belief state* $S = (S_1, \ldots, S_n)$ of M such that each $S_i \in \text{BS}_i$. A bridge rule r of the form (2) is *applicable* in S iff body$(r)^+|_{c_k} \subseteq S_{c_k}$ and body$(r)^-|_{c_k} \cap S_{c_k} = \emptyset$ for all $1 \leq k \leq m$. A belief state $S = (S_1, \ldots, S_n)$ of an MCS $M = (C_1, \ldots, C_n)$ is an *equilibrium* iff $S_i \in \text{ACC}_i(kb_i \cup \{\text{head}(r) \mid r \in br_i,\ r \text{ is applicable in } S\})$ for all $1 \leq i \leq n$.

In practice, however, we are more interested in equilibria of a *subsystem* with root context C_k, e.g., when querying to a context. Naturally, such partial equilibria have to contain coherent information from C_k and all contexts in the import closure of C_k, and therefore, are parts of potential equilibria of the whole system. We define the *import neighbourhood* of a context C_k as the set $In(k) = \{c \mid (c : b) \in \text{body}(r), r \in br_k\}$ and the *import closure* $IC(k)$ as the smallest set of contexts S such that (1) $C_k \in S$ and (2) $C_i \in S$ implies $\{C_j \mid j \in In(i)\} \subseteq S$. Let $\varepsilon \notin \mathcal{A}$ be a new symbol representing the value 'unknown'. A *partial belief state* of M is a sequence $S = (S_1, \ldots, S_n)$, such

that $S_i \in \mathrm{BS}_i \cup \{\varepsilon\}$ for all $1 \leq i \leq n$. A partial belief state $S = (S_1, \ldots, S_n)$ of MCS $M = (C_1, \ldots, C_n)$ w.r.t. C_k is a *partial equilibrium* iff whenever $C_i \in IC(k)$, $S_i \in \mathrm{ACC}_i(kb_i \cup \{\mathrm{head}(r) \mid r \in br_i,\ r \text{ is applicable in } S\})$, otherwise $S_i = \varepsilon$, for all $1 \leq i \leq n$.

Example 1. As a running example, consider the MCS $M = (C_1, C_2, C_3)$ with ASP logics over alphabets $\mathcal{A}_1 = \{a, b, c\}$, $\mathcal{A}_2 = \{d, e, f, g\}$, and $\mathcal{A}_3 = \{h\}$. Suppose

$$kb_1 = \left\{ c \leftarrow a, b, {\sim}c \right\}, \qquad kb_2 = \left\{ \begin{array}{l} f \leftarrow d, e, {\sim}g \\ g \leftarrow d, e, {\sim}f \end{array} \right\}, \quad kb_3 = \emptyset,$$

$$br_1 = \left\{ \begin{array}{l} a \leftarrow {\sim}(2:d) \\ b \leftarrow {\sim}(2:e) \end{array} \right\}, \quad br_2 = \left\{ \begin{array}{l} d \leftarrow {\sim}(1:a) \\ e \leftarrow {\sim}(1:b) \end{array} \right\}, \quad br_3 = \left\{ h \leftarrow (1:a) \right\}.$$

Then, $(\{b\}, \{d\}, \varepsilon)$, $(\{a\}, \{e\}, \varepsilon)$, $(\emptyset, \{d, e, f\}, \varepsilon)$, and $(\emptyset, \{d, e, g\}, \varepsilon)$ are partial equilibria w.r.t. C_1, and $(\{b\}, \{d\}, \emptyset)$, $(\{a\}, \{e\}, \{h\})$, $(\emptyset, \{d, e, f\}, \emptyset)$, and $(\emptyset, \{d, e, g\}, \emptyset)$ are equilibria. Observe that M remains invariant under a swap of atoms f and g, which is what we will call a symmetry of M. Furthermore, the subsystem given by $IC(1) = \{C_1, C_2\}$ remains invariant under a swap of atoms f and g, and/or a simultaneous swap of atoms a, b and d, e, which is what we will call a partial symmetry of M w.r.t. $\{C_1, C_2\}$.

3 Algebraic Background

Intuitively, a symmetry of a discrete object is a transformation of its components that leaves the object unchanged. Symmetries are studied in terms of groups. Recall that a *group* is an abstract algebraic structure $(G, *)$, where G is a set closed under a binary associative operation $*$ such that there is a *unit* element and every element has a unique *inverse*. Often, we abuse notation and refer to the group G, rather than to the structure $(G, *)$, and we denote the size of G as $|G|$. A compact representation of a group is given through generators. A set of group elements such that any other group element can be expressed in terms of their product is called a *generating set* or *set of generators*, and its elements are called *generators*. A generator is *redundant*, if it can be expressed in terms of other generators. A generating set is *irredundant*, if no strict subset of it is generating. Such a set provides an extremely compact representation of a group. In fact, representing a finite group by an irredundant generating set ensures exponential compression, as it contains at most $\log_2|G|$ elements [1].

A mapping $f\colon G \to H$ between two groups $(G, *)$ and $(H, \circ)$ is a *homomorphism* iff for $a, b \in G$ we have that $f(a * b) = f(a) \circ f(b)$; if it has also an inverse that is a homomorphism, f is an *isomorphism*, which is an *automorphism* if $G = H$. The groups G and H are called *isomorphic*, if there exists some isomorphism between them. Any group isomorphism maps (irredundant) generating sets to (irredundant) generating sets [1]. The domain G of f is denoted as $\mathrm{dom}(f)$. In our context, the group of permutations is most important. Recall that a *permutation* of a set S is a bijection $\pi\colon S \to S$. It is well-known that the set of all permutations of S form a group under composition, denoted as $\Pi(S)$.

The image of $a \in S$ under a permutation π is denoted as a^π, and for vectors $s = (a_1, a_2, \ldots, a_k) \in S^k$ define $s^\pi = (a_1^\pi, a_2^\pi, \ldots, a_k^\pi)$. For formulas $\phi(a_1, a_2, \ldots, a_k)$ of some logic over alphabet $\mathcal{A}$ s. t. $S \subseteq \mathcal{A}$ define $\phi^\pi(a_1, a_2, \ldots, a_k) = \phi(a_1^\pi, a_2^\pi, \ldots, a_k^\pi)$,

e.g., for a rule r of form (1), let r^π be $a_1^\pi; \ldots ; a_\ell^\pi \leftarrow b_1^\pi, \ldots, b_j^\pi, {\sim}b_{j+1}^\pi, \ldots, {\sim}b_m^\pi$. For a bridge rule r of form (2) define $r^\pi = a^\pi \leftarrow (c_1 : b_1^\pi), \ldots, (c_j : b_j^\pi), {\sim}(c_{j+1} : b_{j+1}^\pi), \ldots, {\sim}(c_m : b_m^\pi)$. Finally, for a set X (of elements or subsets from S, formulas, bridge rules, etc.), define $X^\pi = \{x^\pi \mid x \in X\}$.

We will make use of the *cycle notation* where a permutation is a product of disjoint cycles. A cycle $(a_1\ a_2\ a_3\ \cdots\ a_n)$ means that the permutation maps a_1 to a_2, a_2 to a_3, and so on, finally a_n back to a_1. An element that does not appear in any cycle is understood as being mapped to itself. The *orbit* of $a \in S$ under a permutation $\pi \in \Pi(S)$ are the set of elements of S to which a can be mapped by (repeatedly) applying π. Note that orbits define an equivalence relation on elements (sets, vectors, etc.) of S.

In graph theory, the symmetries are studied in terms of graph automorphisms. We consider directed graphs $G = (V, E)$, where V is a set of vertices and $E \subseteq V \times V$ is a set of directed edges. Intuitively, an automorphism of G is a permutation of its vertices that maps edges to edges, and non-edges to non-edges, preserving edge orientation. More formally, an *automorphism* or a *symmetry of* G is a permutation $\pi \in \Pi(V)$ such that $(u, v)^\pi \in E$ iff $(u, v) \in E$. An extension considers vertex colourings that are partitionings $\rho(V) = \{V_1, V_2, \ldots, V_k\}$ of the nodes V into disjoint nonempty sets ("colours") V_i. Symmetries must map each vertex to a vertex with the same colour. Formally, given a colouring of the vertices $\rho(V) = \{V_1, V_2, \ldots, V_k\}$, an *automorphism* or a *symmetry of a coloured graph* G is a symmetry π of G s.t. $\rho(V)^\pi = \rho(V)$. The *graph automorphism* problem (GAP) is to find all symmetries of a given graph, for instance, in terms of generators. GAP is not known to be solvable in polynomial time, and its decisional variant is known to be within the complexity classes P and NP, but there is strong evidence that this problem is not NP-complete (cf. [2]). Thus it is potentially easier than, for instance, deciding answer set existence.

4 Symmetry in Multi-Context Systems

We will now define our notion of a symmetry of a multi-context system. In this section we consider MCS $M = (C_1, \ldots, C_n)$ with logics L_i over alphabet $\mathcal{A}_i$, for $1 \leq i \leq n$.

Definition 1. *A* symmetry of M *is a permutation* $\pi \in \Pi(\mathcal{A})$ *such that (1)* $\mathcal{A}_i^\pi = \mathcal{A}_i$*, (2)* $kb_i^\pi = kb_i$*, and (3)* $br_i^\pi = br_i$*, for* $1 \leq i \leq n$*.*

In this definition, items (2) and (3) capture the intention that symmetries are permutations of beliefs which yield identical knowledge bases and bridge rules, respectively. Item (1) imposes that symmetries do not alter the indiviuale context languages; there is no technical need for this, i.e., dropping (1) would yield a more general definition of symmetry for which our subsequent results would still hold; however the respective additional symmetries are irrelevant from a practical point of view and thus disregarded. For the same reason, we disregard permutations of the order of contexts.

Sometimes, a symmetry affects only atoms of a single context, i.e., behaves like the identity for the atoms of all other contexts. A symmetry π of M is *local* for context C_k iff $a^\pi = a$ for all $a \in \mathrm{dom}(\pi) \setminus \mathcal{A}_k$.

Example 2 (cont'd). Reconsider the MCS $M = (C_1, C_2, C_3)$ from Example 1. Symmetries of M are given through the identity and $(f\ g)$, both are local for C_2.

Similar to belief states, we define the notion of partial symmetries, which are parts of potential symmetries of the system.

Definition 2. *A permutation π of the elements in $S \subseteq \mathcal{A}$ is a* partial symmetry *of M w.r.t. the set of contexts $C = \{C_{i_1}, \ldots, C_{i_m}\}$ iff (1) $\mathcal{A}_{i_k} \cup at(br_{i_k}) \subseteq S$ (2) $\mathcal{A}^{\pi}_{i_k} = \mathcal{A}_{i_k}$, (3) $kb^{\pi}_{i_k} = kb_{i_k}$, and (4) $br^{\pi}_{i_k} = br_{i_k}$, for all $1 \leq k \leq m$.*

For combining partial symmetries π and σ, we define their *join* $\pi \bowtie \sigma$ as the permutation θ, where

$$a^{\theta} = \begin{cases} a^{\pi} & \text{if } a \in \text{dom}(\pi), \\ a^{\sigma} & \text{if } a \in \text{dom}(\sigma). \end{cases}$$

whenever $a^{\pi} = a^{\sigma}$ for all $a \in \text{dom}(\pi) \cap \text{dom}(\sigma)$; otherwise, the join is undefined. The *join* of two sets of partial symmetries of M is naturally defined as $\Pi \bowtie \Sigma = \{\pi \bowtie \sigma \mid \pi \in \Pi,\ \sigma \in \Sigma\}$. Note that, $\pi \bowtie \sigma$ is void, i.e., undefined, if π and σ behave different for some $a \in \text{dom}(\pi) \cap \text{dom}(\sigma)$. Otherwise, the join is a partial symmetry of M.

Theorem 1. *Let $M = (C_1, \ldots, C_n)$ be an MCS with logics L_i over alphabet $\mathcal{A}_i$. (1) Every partial symmetry of M w.r.t. $\{C_1, \ldots, C_n\}$ is a symmetry of M. (2) For every partial symmetries π and σ of M w.r.t. $C_{(\pi)} = \{C_{i_1}, \ldots, C_{i_m}\}$ and $C_{(\sigma)} = \{C_{j_1}, \ldots, C_{j_\ell}\}$, respectively, such that $\theta = \pi \bowtie \sigma$ is defined, θ is a partial symmetry of M w.r.t. $C_{(\pi)} \cup C_{(\sigma)}$.*

Proof. (1) Let θ be a partial symmetry of M w.r.t. $\{C_1, \ldots, C_n\}$. By Definition 2 we have $\text{dom}(\theta) \subseteq \bigcup_{i=1}^{n} \mathcal{A}_i = \mathcal{A}$ (an upper bound for the domain of partial symmetries), and $\mathcal{A}_i \subseteq \text{dom}(\pi)$ (lower bound for domain of partial symmetries) for $1 \leq i \leq n$. Hence, θ is a permutation of exactly the elements in $\mathcal{A}$. Given this, and since $\mathcal{A}^{\theta}_i = \mathcal{A}_i$, $kb^{\theta}_i = kb_i$ and $br^{\theta}_i = br_i$ holds for $1 \leq i \leq n$, i.e., all contexts in M, we have that θ is a symmetry of M. (2) We check that all conditions of a partial symmetry hold for θ. By definition of the join, $\text{dom}(\theta) = \text{dom}(\pi) \cup \text{dom}(\sigma) \supseteq \bigcup_{k=1}^{m}(\mathcal{A}_{i_k} \cup \text{at}(br_{i_k})) \cup \bigcup_{k=1}^{\ell}(\mathcal{A}_{j_k} \cup \text{at}(br_{j_k}))$. Furthermore, $\mathcal{A}^{\theta}_{i_k} = \mathcal{A}^{\pi}_{i_k} = \mathcal{A}_{i_k}$, $kb^{\theta}_{i_k} = kb^{\pi}_{i_k} = kb_{i_k}$ and $br^{\theta}_{i_k} = br^{\pi}_{i_k} = br_{i_k}$ for all $1 \leq k \leq m$, and similarly, $\mathcal{A}^{\theta}_{j_k} = \mathcal{A}^{\sigma}_{j_k} = \mathcal{A}_{j_k}$, $kb^{\theta}_{j_k} = kb^{\sigma}_{j_k} = kb_{j_k}$ and $br^{\theta}_{j_k} = br^{\sigma}_{j_k} = br_{j_k}$ for all $1 \leq k \leq \ell$. Hence, θ is a partial symmetry of M w.r.t. $C_{(\pi)} \cup C_{(\sigma)}$. □

Observe that every partial symmetry of M w.r.t. a set of contexts C is a partial symmetry of M w.r.t. a non-empty subset of C; a partial symmetry can always be written as the join of two partial symmetries.

Example 3 (cont'd). Reconsider M from Example 1. The partial symmetries Π of M w.r.t. $\{C_1\}$ are given through the identity id and $(a\ b)\ (d\ e)$. The partial symmetries Σ of M w.r.t. $\{C_2\}$ are given through id, $(a\ b)\ (d\ e)\ (f\ g)$, and $(f\ g)$. The partial symmetries of M w.r.t. $\{C_1, C_2\}$ are $\Pi \bowtie \Sigma = \Sigma$, and the partial symmetries Θ of M w.r.t. $\{C_3\}$ are just id alone. The symmetries of M are $\Pi \bowtie \Theta = \{\text{id}, (f\ g)\}$.

5 Distributed Symmetry Detection

In the following, we provide a distributed algorithm for detecting symmetries of an MCS $M = (C_1, \ldots, C_n)$. We follow Dao-Tran *et al.* [7] by taking a local stance, i.e.,

Algorithm: DSD(H) at context C_k
Input: Visited contexts H.
Data: Cache $c(k)$.
Output: The set of accumulated partial symmetries Π.

if $c(k)$ is not initialised **then** $c(k) \leftarrow \text{LSD}(C_k)$;
$H \leftarrow H \cup \{k\}$;
$\Pi \leftarrow c(k)$;
foreach $i \in In(k) \setminus H$ **do** $\Pi \leftarrow \Pi \bowtie C_i.\text{DSD}(H)$;
return Π;

Fig. 1. The distributed symmetry detection algorithm.

we consider a context C_k and those parts of the system that are in the import closure of C_k to compute (potential) symmetries of the system. To this end, we design an algorithm whose instances run independently at each context node and communicate with other instances for exchanging sets of partial symmetries. This provides a method for distributed symmetry building.

The idea is as follows: starting from a context C_k, we visit the import closure of C_k by expanding the import neighbourhood at each context, maintaining the set of visited contexts in a set H, the *history*, until a leaf context is reached, or a cycle is detected by noticing the presence of a neighbour context in H. A leaf context C_i simply computes all partial symmetries of M w.r.t. $\{C_i\}$. Then, it returns the results to its parent (the invoking context), for instance, in form of permutation cycles. The results of intermediate contexts C_i are partial symmetries of M w.r.t. $\{C_i\}$, which can be joined, i.e., consistently combined, with partial symmetries from their neighbours, and resulting in partial symmetries of M w.r.t. $IC(i)$. In particular, the starting context C_k returns its partial symmetries joined with the results from its neighbours, as a final result. We assume that each context C_k has a background process that waits for incoming requests with history H, upon which it starts the computation outlined in our algorithm shown in Fig. 1. We write $C_i.\text{DSD}(H)$ to specify that we send H to the process at context C_i and wait for its return message. This process also serves the purpose of keeping the cache $c(k)$ persistent. We use the primitive $\text{LSD}(C_k)$ which computes all partial symmetries of M w.r.t. $\{C_k\}$ over $\mathcal{A}_k \cup \text{at}(br_k)$.

Our algorithm proceeds in the following way:

1. Check the cache for partial symmetries of M w.r.t. $\{C_k\}$;
2. if imports from neighbour contexts are needed, then request partial symmetries from all neighbours and join them (previously visited contexts excluded). This can be performed in parallel. Also, partial symmetries can be joined in the order neighbouring contexts do answer; and
3. return partial symmetries of M w.r.t. $IC(k)$.

Correctness of our approach hold by the following result.

Theorem 2. *Let $M = (C_1, \ldots, C_n)$ be an MCS and C_k be a context in M. Then, $\pi \in C_k.\text{DSD}(\emptyset)$ iff π is a partial symmetry of M w.r.t. $IC(k)$.*

Proof (sketch). ($\Rightarrow$) We prove soundness, i.e., if $\pi \in C_k.\text{DSD}(\emptyset)$ then π is a partial symmetry of M w.r.t. $IC(k)$. We proceed by structural induction on the topology of an MCS, and start with acyclic MCS M. Base case: C_k is a leaf with $br_k = \emptyset$ and

$In(k) = \emptyset$. By assumption, LSD(C_k) computes all partial symmetries of M w.r.t. $\{C_k\}$, i.e., $c(k) \leftarrow$ LSD(C_k) in the algorithm in Fig. 1. Induction step: for non-leaf C_k, suppose $In(k) = \{i_1, \ldots, i_m\}$ and $\Pi_k =$ LSD(C_k), $\Pi_{i_j} = C_{i_j}$.DSD($H \cup \{k\}$) for $1 \leq j \leq m$. By Theorem 1, $\Pi = \Pi_k \bowtie \Pi_{i_1} \bowtie \cdots \bowtie \Pi_{i_m}$, as computed by $\Pi \leftarrow \Pi \bowtie C_i$.DSD($H$) in the loop of the algorithm in Fig. 1, consists of partial symmetries of M w.r.t. $IC(k)$.

The proof for cyclic M is similar. In a run we eventually end up in C_i such that $i \in H$ again. In that case, calling C_i.DSD(H) is discarded, which breaks the cycle. However, partial symmetries excluding C_i are propagated through the system to the calling C_i which combines the intermediate results with partial symmetries of M w.r.t. $\{C_i\}$.
($\Leftarrow$) We give now a proof sketch for completeness. Let π be a partial symmetry of M w.r.t. $IC(k)$. We show $\pi \in C_k$.DSD($\emptyset$). The proof idea is as follows: we proceed as in the soundness part by structural induction on the topology of M, and in the base case for a leaf context C_k, by assumption, we get that LSD(C_k) returns all partial symmetries of M w.r.t. $\{C_k\}$, i.e., all partial symmetries of M w.r.t. $IC(k)$. For the induction step, we verify straightforward that π being a partial symmetry of M w.r.t. $IC(k)$ implies π being a partial symmetry of M w.r.t. $IC(i)$ for all $i \in In(k)$. □

6 Symmetry Detection via Graph Automorphism

The primitive LSD(C_i) for detecting partial symmetries of an MCS $M = (C_1, \ldots, C_n)$ w.r.t. $\{C_i\}$ using logic L_i has to be defined for every logic L_i anew. As an example, our approach for detecting partial symmetries of M w.r.t. an ASP context C_i is through reduction to, and solution of, an associated graph automorphism problem. The graph $GAP(C_i)$ is constructed as follows:

1. Every atom that occurs in $kb_i \cup br_i$ (every context atom $(c : b)$ in br_i, respectively) is represented by two vertices of colour i (c, respectively) and $n + 1$ that correspond to the positive and negative literals.
2. Every rule (every bridge rule, respectively) is represented by a *body vertex* of colour $n + 2$ ($n + 3$, respectively), a set of directed edges that connect the vertices of the literals (context literals, respectively) that appear in the rule's body to its body vertex, and a set of directed edges that connect the body vertex to the vertices of the atoms that appear in the head of the rule.
3. To properly respect negation, that is, an atom a maps to b if and only if $\sim a$ maps to $\sim b$ for any atoms a and b, vertices of opposite (context) literals are mated by a directed edge from the positive (context) literal to the negative (context) literal.

Example 4 (cont'd). Reconsider MCS M from Example 1. Fig. 2 illustrates $GAP(C_2)$, where different shapes and tones represent different colours.

Symmetries of $GAP(C_i)$ correspond precisely to the partial symmetries of M w.r.t. $\{C_i\}$.

Theorem 3. *Let $M = (C_1, \ldots, C_n)$ be an MCS with ASP context C_i. The partial symmetries of M w.r.t. $\{C_i\}$ correspond one-to-one to the symmetries of $GAP(C_i)$.*

Proof. The proof for logic programs is shown in [8]. Therefore we only provide arguments regarding bridge rules and context atoms. ($\Rightarrow$) A partial symmetry of M

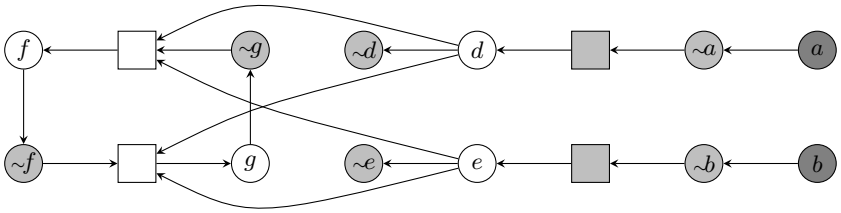

Fig. 2. GAP reduction of context C_2 from Example 1.

w.r.t. $\{C_i\}$ will map context atoms to context atoms of the same context. Since they have the same colour, the symmetry is preserved for corresponding vertices and consistency edges. The same applies to body vertices and edges representing bridge rules, since the body vertices have incoming edges from context literal vertices with their respective colour only, and vertices of the same colour are mapped one to another. Thus, a consistent mapping of atoms in C_k, when carried over to the graph, must preserve symmetry. ($\Leftarrow$) We now show that every symmetry in the graph corresponds to a partial symmetries of M w.r.t. $\{C_i\}$. Recall that we use one colour for positive context literals from each context, one for negative context literals from each context, and one for bodies. Hence, a graph symmetry must map (1) positive context literal vertices to other such from the same context, negative literal vertices to negative literal vertices from the same context, and body vertices to body vertices, and (2) the body edges of a vertex to body edges of its mate. This is consistent with partial symmetries of M w.r.t. $\{C_i\}$ mapping context atoms to context atoms, and bodies to bodies, i.e., bridge rules to bridge rules. □

Corollary 1. *Let $M = (C_1, \ldots, C_n)$ be an MCS with ASP context C_i. The partial symmetry group of M w.r.t. $\{C_i\}$ and the symmetry group of $GAP(C_i)$ are isomorphic. Furthermore, sets of partial symmetry generators of M w.r.t. $\{C_i\}$ correspond one-to-one to sets of symmetry generators of $GAP(C_i)$.*

To detect local symmetries only, we further modify our approach by assigning a unique colour to each context atom and each atom that is referenced in other contexts, i.e., context atoms cannot be mapped.

With reference to related work (cf. [1, 8]), we stretch that the detection of symmetries through reduction to graph automorphism is computationally quite feasible, i.e., the overhead cost in situations that do not have symmetries is negligible.

7 Distributed Symmetry-breaking Constraints

Recall that a (partial) symmetry of an MCS defines equivalence classes on its (partial) equilibria through orbits. Symmetry breaking amounts to selecting some representatives from every equivalence class and formulating conditions, composed into a (distributed) symmetry-breaking constraint (SBC), that is only satisfied on those representatives. A *full* SBC selects exactly one representative from each orbit, otherwise we call an SBC *partial*. The most common approach is to order all elements from the solution space lexicographically, and to select the lexicographically smallest element, the *lex-leader*,

from each orbit as its representative (see, for instance, [1, 6]). A *lex-leader symmetry-breaking constraint* (LL-SBC) is an SBC that is satisfied only on the lex-leaders of orbits. Given an MCS $M = (C_1, \ldots, C_n)$ with logics L_i over alphabet $\mathcal{A}_i$, we will assume a total ordering $<_{\mathcal{A}}$ on the atoms $a_1, a_2, \ldots, a_m$ in $\mathcal{A}$ and consider the induced lexicographic ordering on the (partial) belief states. Following [6], we obtain an LL-SBC by encoding a (distributed) *permutation constraint* (PC) for every permutation π, where

$$\mathrm{PC}(\pi) = \bigwedge_{1 \leq i \leq m} \left[\bigwedge_{1 \leq j \leq i-1} (a_j = a_j^\pi) \right] \rightarrow (a_i \leq a_i^\pi).$$

By *chaining*, which uses atoms $c_{\pi,i}$, $1 < i \leq m+1$ (which informally express that for some $i \leq j \leq m$ the implication fails if it did not for some $j < i$), we achieve a representation that is linear in the number of atoms [1]:

$$\begin{aligned} \mathrm{PC}(\pi) \;&=\; (a_1 \leq a_1^\pi) \wedge \neg c_{\pi,2}, \\ \neg c_{\pi,i} &\leftrightarrow ((a_{i-1} \geq a_{i-1}^\pi) \rightarrow (a_i \leq a_i^\pi) \wedge c_{\pi,i+1}) \qquad 1 < i \leq m, \\ \neg c_{\pi,m+1} &\leftrightarrow \top. \end{aligned}$$

In order to distribute the PC formula in M, given the total ordering $<_{\mathcal{A}}$, we define (the truth of) atoms $c_{\pi,i}$ in the contexts C_k such that $a_{i-1} \in \mathcal{A}_k$. Observe that, for each subformula, the atoms a_i, a_i^π and $c_{\pi,i+1}$ might be defined in a different context j, and their truth value has to be imported via bridge rules. We thus introduce auxiliary atoms a'_i, $a_i^{'\pi}$, and $c'_{\pi,i+1}$ in C_k that resemble the truth of a_i, a_i^π, and $c_{\pi,i+1}$, respectively. Then we distribute $\mathrm{PC}(\pi)$ to each context C_k for each $1 \leq k \leq n$ as follows:

$$\begin{aligned} \mathrm{PC}(\pi) \;&=\; (a_1 \leq a_1^\pi) \wedge \neg c_{\pi,2} && \text{if } a_1 \in \mathcal{A}_k, \\ \neg c_{\pi,i} &\leftrightarrow ((a_{i-1} \geq a_{i-1}^\pi) \rightarrow (a_i \leq a_i^\pi) \wedge \neg c_{\pi,i+1}) && \text{if } a_{i-1}, a_i \in \mathcal{A}_k, \\ \neg c_{\pi,i} &\leftrightarrow ((a_{i-1} \geq a_{i-1}^\pi) \rightarrow (a'_i \leq a_i^{'\pi}) \wedge \neg c'_{\pi,i+1}) && \text{if } a_{i-1} \in \mathcal{A}_k, a_i \in \mathcal{A}_j, j \neq k, \\ \neg c_{\pi,m+1} &\leftrightarrow \top && \text{if } a_m \in \mathcal{A}_k, \\ \\ a'_i &\leftarrow (j : a_i) && \text{if } a_{i-1} \in \mathcal{A}_k, a_i \in \mathcal{A}_j, j \neq k, \\ a_i^{'\pi} &\leftarrow (j : a_i^\pi) && \text{if } a_{i-1} \in \mathcal{A}_k, a_i \in \mathcal{A}_j, j \neq k, \\ c'_{\pi,i+1} &\leftarrow (j : c_{\pi,i+1}) && \text{if } a_{i-1} \in \mathcal{A}_k, a_i \in \mathcal{A}_j, j \neq k. \end{aligned}$$

The distributed PC can be adjusted to other logics as well. Exploiting detected symmetries has been studied, e.g., in the context of SAT [1, 6], planning [9], and constraint programming [15]. For an ASP context C_k, we can express the distributed PC as follows:

$$\left.\begin{aligned} &\leftarrow a_1, {\sim}a_1^\pi \\ &\leftarrow c_{\pi,2} \end{aligned}\right\} \text{ if } a_1 \in \mathcal{A}_k; \qquad \left.\begin{aligned} c_{\pi,i} &\leftarrow a_{i-1}, a'_i, {\sim}a_i^{'\pi} \\ c_{\pi,i} &\leftarrow {\sim}a_{i-1}^\pi, a'_i, {\sim}a_i^{'\pi} \\ c_{\pi,i} &\leftarrow a_{i-1}, c'_{\pi,i+1} \\ c_{\pi,i} &\leftarrow {\sim}a_{i-1}^\pi, c'_{\pi,i+1} \end{aligned}\right\} \begin{aligned} &\text{if } a_i \in \mathcal{A}_j, \\ &a_{i-1} \in \mathcal{A}_k, \\ &j \neq k; \end{aligned}$$

$$\left.\begin{aligned} c_{\pi,i} &\leftarrow a_{i-1}, a_i, {\sim}a_i^\pi \\ c_{\pi,i} &\leftarrow {\sim}a_{i-1}^\pi, a_i, {\sim}a_i^\pi \\ c_{\pi,i} &\leftarrow a_{i-1}, c_{\pi,i+1} \\ c_{\pi,i} &\leftarrow {\sim}a_{i-1}^\pi, c_{\pi,i+1} \end{aligned}\right\} \begin{aligned} &\text{if } a_i \in \mathcal{A}_k, \\ &a_{i-1} \in \mathcal{A}_k; \end{aligned} \qquad \left.\begin{aligned} a'_i &\leftarrow (j : a_i) \\ a_i^{'\pi} &\leftarrow (j : a_i^\pi) \\ c'_{\pi,i+1} &\leftarrow (j : c_{\pi,i+1}) \end{aligned}\right\} \begin{aligned} &\text{if } a_i \in \mathcal{A}_j, \\ &a_{i-1} \in \mathcal{A}_k, \\ &j \neq k; \end{aligned}$$

Here, $c_{\pi,i}$ is defined from $\neg c_{\pi,i} \leftrightarrow (\alpha \rightarrow \beta \wedge \neg c_{\pi,i+1})$ via $c_{\pi,i} \leftrightarrow (\alpha \wedge \neg\beta \vee \alpha \wedge c_{\pi,i+1})$ exploiting Clark completion and splitting $\alpha = a_{i-1} \leq a_{i-1}^\pi$ into the (overlapping) cases where a_{i-1} is true and a_{i-1}^π is false. We collect the newly introduced formulas in $kb_{k,\pi}$ and bridge rules in $br_{k,\pi}$ for each $1 \leq k \leq n$. The following correctness result can be shown, generalizing a similar result for ASP programs in [8].

Theorem 4. *Let π be a (partial) symmetry of an MCS $M = (C_1, \ldots, C_n)$ with ASP contexts C_i. A (partial) equilibrium of M satisfies PC(π) iff it is a (partial) equilibrium of $M(\pi) = (C_1(\pi), \ldots, C_n(\pi))$, where $C_k(\pi)$ extends C_k by $kb_k(\pi) = kb_k \cup kb_{k,\pi}$ and $br_k(\pi) = br_k \cup br_{k,\pi}$.*

This result generalizes to MCS having contexts C_i with (possibly heterogeneous) logics L_i that permit to encode PC via additional formulas in the knowledge base kb_i.

Example 5 (cont'd). Reconsider M from Example 1. Given the ordering $a <_{\mathcal{A}} b <_{\mathcal{A}} d <_{\mathcal{A}} e$, the permutation constraint to break the partial symmetry $\pi = (a\ b)\ (d\ e)$ is:

$$\left.\begin{array}{ll} & c_{\pi,2} \leftarrow b, d', {\sim}e' \\ \leftarrow a, {\sim}b & c_{\pi,2} \leftarrow {\sim}a, d', {\sim}e' \\ \leftarrow c_{\pi,2} & c_{\pi,2} \leftarrow b, c'_{\pi,3} \\ & c_{\pi,2} \leftarrow {\sim}a, c'_{\pi,3} \end{array}\right\} kb_{1,\pi}, \qquad \left.\begin{array}{c} d' \leftarrow (2:d) \\ e' \leftarrow (2:e) \\ c'_{\pi,3} \leftarrow (2:c_{\pi,3}) \end{array}\right\} br_{1,\pi}, \text{ and}$$

$kb_{2,\pi} = br_{2,\pi} = \emptyset$. One can check that $(\{b\}, \{e\}, \varepsilon)$, $(\emptyset, \{d, e, f\}, \varepsilon)$, and $(\emptyset, \{d, e, g\}, \varepsilon)$ are partial equilibria of $M(\pi)$ w.r.t C_1, and $(\{a\}, \{d\}, \varepsilon)$ is not (cf. Example 1) since $(\{a\}, \{d\}, \varepsilon) <_{\mathcal{A}} (\{b\}, \{e\}, \varepsilon)$.

The LL-SBC that breaks every (partial) symmetry in an MCS, denoted LL-SBC(Π), can now be constructed by conjoining all of its permutation constraints [6]. We can add LL-SBC(Π) to M, say $M(\Pi) = (C_1(\Pi), \ldots, C_n(\Pi))$, where $C_k(\Pi)$ extends C_k by $kb_k(\Pi) = kb_k \cup \bigcup_{\pi \in \Pi} kb_k(\pi)$ and $br_k(\Pi) = br_k \cup \bigcup_{\pi \in \Pi} br_k(\pi)$.

Breaking all symmetries may not speed up search because there are often exponentially many of them. A better trade-off may be provided by breaking enough symmetries [6]. We explore partial SBCs, i.e., we do not require that SBCs are satisfied by lex-leading assignments only (but we still require that all lex-leaders satisfy SBCs). Irredundant generators are good candidates because they cannot be expressed in terms of each other, and implicitly represent all symmetries. Hence, breaking all symmetry in a generating set can eliminate all problem symmetries.

8 Experiments

We present some results on breaking local symmetries in terms of irredundant generators for distributed nonmonotonic MCS with ASP logics. Experiments consider the DMCS system [7] and its optimized version DMCSOPT [3]. Both systems are using the ASP solver CLASP [10] as their core reasoning engine. However, in contrast to DMCS, DMCSOPT exploits the topology of an MCS, that is the graph where contexts are nodes and import relations define edges, using decomposition techniques and minimises communication between contexts by projecting partial belief states to relevant atoms. We compare the average response time and the number of solutions under symmetry breaking, denoted as DMCS$^{\pi}$ and DMCSOPT$^{\pi}$, respectively, on benchmarks versus direct application of the respective systems. All tests were run on a 2×1.80 GHz PC under Linux, where each run was limited to 180 seconds. Our benchmarks stem from [3] and include random MCSs with various fixed topologies that should resemble the context dependencies of realistic scenarios. Experiments consider MCS instances with ordinary (D) and zig-zag (Z) diamond stack, house stack (H), and ring (R). A diamond stack combines multiple diamonds

Table 1. Completed runs (10 random instances each): avg. running time (secs) vs. timeouts

		DMCS		DMCS$^\pi$		DMCSOPT		DMCSOPT$^\pi$	
	n	time	#t.out	time	#t.out	time	#t.out	time	#t.out
D	10	1.90		0.46		0.54		0.35	
	13	62.12	4	32.21	2	1.38		0.98	
	25	—	10	—	10	16.12		11.72	
	31	—	10	—	10	84.02	1	58.95	
H	9	7.54		1.89		0.33		0.20	
	13	88.85	6	63.98	2	0.60		0.35	
	41	—	10	—	10	1.38		0.95	
	101	—	10	—	10	5.48		3.58	
R	10	0.36		0.26		0.15		0.12	
	13	22.41	1	5.11		0.19		0.16	
Z	10	6.80		3.24		0.62		0.37	
	13	57.58	3	42.93	3	1.03		0.68	
	70	—	10	—	10	18.87		9.98	
	151	—	10	—	10	51.10		30.15	

in a row, where ordinary diamonds (in contrast to zig-zag diamonds) have no connection between the 2 middle contexts. A house consists of 5 nodes with 6 edges such that the ridge context has directed edges to the 2 middle contexts, which form with the 2 base contexts a cycle with 4 edges. House stacks are subsequently built using the basement nodes as ridges for the next houses.

Table 1 shows some experimental results on calculating equilibria w.r.t. a randomly selected starting context of MSC with n contexts, where n varies between 9 and 151. Each context has an alphabet of 10 atoms, exports at most 5 atoms to other contexts, and has a maximum of 5 bridge rules with at most 2 bridge literals. First, we confirm the results of Bairakdar *et al.* [3], i.e., DMCSOPT can handle larger sizes of MCSs more efficiently than DMCS. Second, evaluating the MCS instances with symmetry breaking compared to the direct application of either DMCS or DMCSOPT yields improvements in response time throughout all tested topologies. In fact, symmetry breaking always leads to better runtimes, and in some cases, returns solutions to problems which are otherwise intractable within the given time.

Fig. 3 presents the average compression of the solution space achieved by symmetry breaking. While the results for DMCS$^\pi$ range between 45% and 80%, the impact of symmetry breaking within DMCSOPT on the number of solutions varies between 5% and 65%. We explain the latter with the restriction of DMCSOPT to relevant atoms defined by the calling context.

9 Conclusion

We have presented a method for distributed symmetry detection and breaking for MCS. In particular, we have designed a distributed algorithm such that each context computes its own (partial) symmetries and communicates them with another for exchanging partial symmetries in order to compute symmetries of the system as a whole. Distributed symmetry-breaking constraints prevent an evaluation engine from ever visiting two points

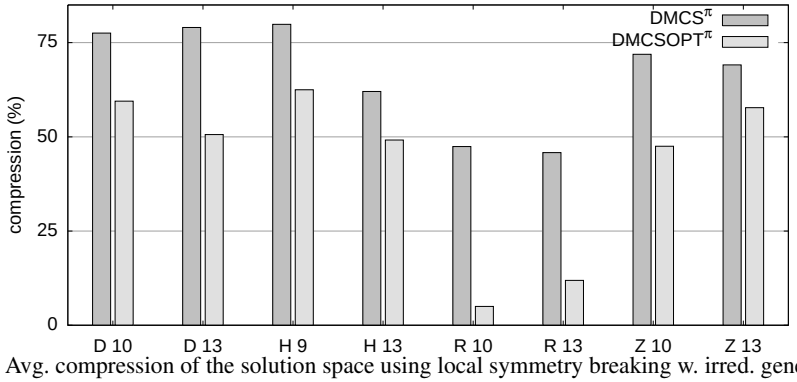

Fig. 3. Avg. compression of the solution space using local symmetry breaking w. irred. generators.

in the search space that are equivalent under the symmetry they represent. We have instantiated symmetry detection and symmetry breaking for MCS with ASP contexts, i.e., we have reduced partial symmetry of an ASP context to the automorphism of a coloured graph and encode symmetry breaking constraints as a distributed logic program. Experiments on recent MCS benchmarks and show promising results. Future work concerns a join operator for partial symmetries that preserves irredundant generators.

References

1. Aloul, F.A., Markov, I.L., Sakallah, K.A.: Shatter: efficient symmetry-breaking for Boolean satisfiability. In: DAC'03. pp. 836–839. ACM (2003)
2. Babai, L.: Automorphism groups, isomorphism, reconstruction. In: Graham, R.L., Grötschel, M., Lovász, L. (eds.) Handbook of Combinatorics, vol. 2, pp. 1447–1540. Elsevier (1995)
3. Bairakdar, S., Dao-Tran, M., Eiter, T., Fink, M., Krennwallner, T.: Decomposition of distributed nonmonotonic multi-context systems. In: JELIA'10. pp. 24–37. Springer (2010)
4. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
5. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: AAAI'07. pp. 385–390. AAAI Press (2007)
6. Crawford, J., Ginsberg, M., Luks, E., Roy, A.: Symmetry-breaking predicates for search problems. In: KR'96. pp. 148–159. Morgan Kaufmann (1996)
7. Dao-Tran, M., Eiter, T., Fink, M., Krennwallner, T.: Distributed nonmonotonic multi-context systems. In: KR'10. pp. 60–70. AAAI Press (2010)
8. Drescher, C., Tifrea, O., Walsh, T.: Symmetry-breaking answer set solving (2011), to appear
9. Fox, M., Long, D.: The detection and exploitation of symmetry in planning problems. In: IJCAI'99. pp. 956–961. Morgan Kaufmann (1999)
10. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: clasp: A conflict-driven answer set solver. In: LPNMR'07. pp. 260–265. Springer (2007)
11. Giunchiglia, F.: Contextual reasoning. Epistemologia, special issue on I Linguaggi e le Macchine 345, 345–364 (1992)
12. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics or: How we can do without modal logics. Artif. Intell. 65(1), 29–70 (1994)
13. Katsirelos, G., Narodytska, N., Walsh, T.: Breaking generator symmetry. In: SymCon'09
14. McCarthy, J.: Generality in artificial intelligence. Commun. ACM 30, 1030–1035 (1987)
15. Puget, J.-F.: Automatic detection of variable and value symmetries. In: CP'05. pp. 475–489. Springer (2005)