On the Reification of Global Constraints

Nicolas Beldiceanu
TASC team (INRIA/CNRS), Mines de Nantes, FR-44307 Nantes, France
Nicolas.Beldiceanu@mines-nantes.fr

Mats Carlsson
SICS, P.O. Box 1263, SE-164 29 Kista, Sweden
Mats.Carlsson@sics.se

Pierre Flener and Justin Pearson
Uppsala University, Department of Information Technology, Box 337, SE-751 05 Sweden
Pierre.Flener@it.uu.se, Justin.Pearson@it.uu.se

SICS Technical Report T2012:02
ISSN: 1100-3154

Abstract: We introduce a simple idea for deriving reified global constraints in a systematic way. It is based on the observation that most global constraints can be reformulated as a conjunction of pure functional dependency constraints together with a constraint that can be easily reified. We first show how the core constraints of the Global Constraint Catalogue can be reified and we then identify several reification categories that apply to at least 82% of the constraints in the Global Constraint Catalogue.

Keywords: Global constraint; reification; functional dependency.

February 15, 2012

Contents

1 Introduction

Conventional wisdom has it that many global constraints cannot be easily reified, i.e., augmented with a 0-1 variable reflecting whether the constraint is satisfied (value 1) or not (value 0). Reified constraints are useful for expressing propositional formulas over constraints (e.g., negation, disjunction [19], implication) and for expressing that a certain number of constraints hold (e.g., the cardinality operator [18]). Using known algorithms from automata theory, we have shown [5, page 271: see keyword "reified automaton constraint"] how to reify a global constraint that can be expressed in terms of a finite automaton that does not use any counters [4, 15]. However, many global constraints, such as ALLDIFFERENT [16] and CUMULATIVE [1], cannot be expressed by an automaton whose size is polynomial in the number of variables of the constraint. The importance of the negation of global constraints has recently increased, both in the context of a constraint seeker with negative samples [7] and for proving the equivalence of two constraint models [2, 13].

Many early constraint programming systems, such as CHIP, GNU Prolog, Ilog Solver, and SICStus Prolog, provide reification for arithmetic constraints. For CHIP [10, page 174] this was indirectly done by a construct of the form `if Cond then Pred1 else Pred2` where `Cond` is a binary constraint, while `Pred1` and `Pred2` are program clauses. However, when global constraints started to get introduced (e.g., ALLDIFFERENT and CUMULATIVE), reification was not available for global constraints. We believe that, in the early 1990s, reification was not considered for global constraints since it was believed that reification could only be obtained by modifying the filtering algorithms attached to each global constraint.

2 How to Derive Reified Global Constraints

A global constraint $GC(\mathcal{A})$ can be defined by restrictions $R(\mathcal{A})$ on its arguments $\mathcal{A}$, e.g., restrictions on the bounds of its arguments, and by a condition $C(\mathcal{A})$ on its arguments, i.e., we have $GC(\mathcal{A}) \equiv R(\mathcal{A}) \wedge C(\mathcal{A})$. For instance, for a constraint defined by a finite automaton (e.g., GLOBAL_CONTIGUITY [6, page 1058]), a typical restriction is that the variables take values in a given alphabet (e.g., values 0 and 1 for GLOBAL_CONTIGUITY). See [6, pages 9–17] for other examples of such restrictions. Note that the set of restrictions may be empty, that is $R(\mathcal{A})$ may be always satisfied. We define the *reified version* of $GC(\mathcal{A})$ as $R(\mathcal{A}) \wedge (C(\mathcal{A}) \Leftrightarrow b)$, where b is a 0-1 variable reflecting whether constraint $GC(\mathcal{A})$ holds or not. The motivation for this definition is that the negation of a global constraint $GC(\mathcal{A})$ should still satisfy the restrictions $R(\mathcal{A})$.

Let a *core reifiable constraint* be a constraint of the form of a Boolean combination of linear arithmetic equalities and inequalities and 0-1 variables. We assume that such constraints are already reifiable, without resorting to the methods being developed in this report. This is the case in all constraint programming systems that we are aware of.

We call a *pure functional dependency constraint* (PFD) a constraint where no additional condition is imposed by the constraint other than determining some of its variables; it can therefore never fail when the variables to be determined are unrestricted. For instance, NVALUE [6, page 1466] is a PFD constraint since it just determines the number of distinct values of a collection of variables, while CYCLE$(nc, \langle s_1, \dots, s_n \rangle)$ [6, page 828] is not since it does not hold for all combinations of $\langle s_1, \dots, s_n \rangle$ in $[1, n]$: it determines the number nc of permutation cycles of the sequence $\langle s_1, \dots, s_n \rangle$; as a necessary condition, $\langle s_1, \dots, s_n \rangle$ must be all distinct (note that ALLDIFFERENT is not a PFD constraint). A PFD constraint may determine more than a single variable, witness the SORT [6, page 1772] constraint. The global constraint catalogue [6] contains a significant number (23%) of PFD constraints.

We now provide the key observation that allows us to reify most global constraints in a straightforward way. Given a global constraint $GC(\mathcal{A})$ defined by $R(\mathcal{A}) \wedge C(\mathcal{A})$, it turns out that the condition $C(\mathcal{A})$ can often be reformulated as a conjunction $CF_1(\mathcal{A}_1, \mathcal{V}_1) \wedge \cdots \wedge CF_p(\mathcal{A}_p, \mathcal{V}_p) \wedge CN(\mathcal{A}_{p+1})$ of constraints, where each constraint $CF_i(\mathcal{A}_i, \mathcal{V}_i)$ (with $1 \leq i \leq p$) is a PFD constraint for which the determined variables $\mathcal{V}_i$ do not occur in $\mathcal{A}$, and where $CN(\mathcal{A}_{p+1})$ is a constraint for which reification can be obtained in a known way (i.e., we may use $CN(\mathcal{A}_{p+1}) \Leftrightarrow b$). The arguments of the constraints $CF_i(\mathcal{A}_i, \mathcal{V}_i)$ (with $1 \leq i \leq p$) and $CN(\mathcal{A}_{p+1})$ must obey the following conditions:

- $\mathcal{V}_i$ (with $1 \leq i \leq p$) is a non-empty set of distinct unrestricted variables, i.e., it has an empty intersection with $\mathcal{A} \cup \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_{i-1} \cup \mathcal{V}_{i+1} \cup \cdots \cup \mathcal{V}_p$.

- $\mathcal{A}_i \subseteq \mathcal{A} \cup \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_{i-1}$ (with $1 \leq i \leq p$), i.e., $\mathcal{A}_i$ gets fixed when $\mathcal{A}, \mathcal{V}_1, \ldots, \mathcal{V}_{i-1}$ are fixed.
- $\mathcal{A}_{p+1}$ has a non-empty intersection with $\mathcal{V}_1 \cup \cdots \cup \mathcal{V}_p$ and is included in $\mathcal{A} \cup \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_p$.
- $\mathcal{V}_i$ has a non-empty intersection with $\mathcal{A}_{i+1} \cup \cdots \cup \mathcal{A}_{p+1}$, i.e., each introduced variable is used at least once.

If all the variables of $\mathcal{A}$ that occur in one of the $\mathcal{A}_i$ (with $1 \leq i \leq p$) are fixed, then all variables in $\mathcal{V}_i$ (with $1 \leq i \leq p$) are also fixed, by the PFD constraints. Note that, from the first two conditions, the conjunction $CF_1(\mathcal{A}_1, \mathcal{V}_1) \wedge \cdots \wedge CF_p(\mathcal{A}_p, \mathcal{V}_p)$ never fails when the variables of $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_p$ are unrestricted, i.e., it determines the variables of $\mathcal{V}_1, \ldots, \mathcal{V}_p$ from the arguments $\mathcal{A}$.

In this context, the reified version of $GC(\mathcal{A})$ is expressed as follows:

$$R(\mathcal{A}) \wedge CF_1(\mathcal{A}_1, \mathcal{V}_1) \wedge \cdots \wedge CF_p(\mathcal{A}_p, \mathcal{V}_p) \wedge (CN(\mathcal{A}_{p+1}) \Leftrightarrow b)$$

3 Reification of Core Global Constraints

We now illustrate our approach on the *core* [6, page 199] constraints of the Global Constraint Catalogue, showing how to reify them by using a conjunction of PFD constraints and a constraint for which reification is directly available. Without loss of generality, we ignore the argument restrictions on these constraints. Note that the core constraints ELEMENT, GLOBAL_CARDINALITY, GLOBAL_CARDINALITY_WITH_COSTS, NVALUE, and SORT are PFD constraints and can thus be used in the reformulations of the other core constraints.

ALLDIFFERENT($\langle v_1, \ldots, v_n \rangle$) [6, page 434] is reified as follows:

$$\textsc{sort}(\langle v_1, \ldots, v_n \rangle, \langle w_1, \ldots, w_n \rangle) \wedge (w_1 < w_2 \wedge \cdots \wedge w_{n-1} < w_n) \Leftrightarrow b$$

GLOBAL_CARDINALITY($\langle x_1, \ldots, x_n \rangle, \langle v_1\ o_1, \ldots, v_m\ o_m \rangle$) [6, page 1034], where v_j and o_j (with $j \in [1, m]$) respectively denote the value for which we count the number of occurrences and the corresponding number of occurrences among the x_i variables (with $i \in [1, n]$), is reified as follows:

$$\textsc{global\_cardinality}(\langle x_1, \ldots, x_n \rangle, \langle v_1\ p_1, \ldots, v_m\ p_m \rangle) \wedge \\ (o_1 = p_1 \wedge \cdots \wedge o_m = p_m) \Leftrightarrow b$$

Being a PFD constraint, GLOBAL_CARDINALITY is used in the PFD part of its reformulation, but with *other* determined variables; the reified-constraint part of the reformulation compares the two sets of determined variables.

GLOBAL_CARDINALITY_WITH_COSTS($\langle x_1, \ldots, x_n \rangle, \langle v_1\ o_1, \ldots, v_m\ o_m \rangle, \mathit{matrix}, \mathit{cost}$) [6, page 1052], where the first two arguments have the same meaning as in GLOBAL_CARDINALITY, and *matrix* and *cost* respectively denote a matrix providing the cost of assigning value v_j (with $j \in [1, m]$) to variable x_i (with $i \in [1, n]$) and the sum of the costs of assigning $x_1, \ldots, x_n$, is reified as follows:

$$\textsc{global\_cardinality\_with\_costs}(\langle x_1, \ldots, x_n \rangle, \langle v_1\ p_1, \ldots, v_m\ p_m \rangle, \mathit{matrix}, c) \wedge \\ (o_1 = p_1 \wedge \cdots \wedge o_m = p_m \wedge \mathit{cost} = c) \Leftrightarrow b$$

ELEMENT($i, \langle t_1, \ldots, t_n \rangle, v$) [6, page 958] is reified as follows:

$$\textsc{element}(i, \langle t_1, \ldots, t_n \rangle, w) \wedge (v = w) \Leftrightarrow b$$

The reification involves only an equality constraint because ELEMENT is a PFD constraint, where the only condition is that value v is uniquely determined by the index i and the table $\langle t_1, \ldots, t_n \rangle$.

CUMULATIVE($\langle s_1\ d_1\ e_1\ r_1, \ldots, s_n\ d_n\ e_n\ r_n \rangle, \mathit{limit}$) [6, page 786], where s_i, d_i, e_i, and r_i (with $i \in [1, n]$) respectively denote the *start*, *duration*, *end*, and *resource consumption* of task i, can be reified by a reformulation that uses PFD constraints for determining the maximum resource consumption:

- For each pair of tasks i, j (with $i, j \in [1, n]$) we create a variable r_{ij}, which is the resource consumption of task j if task j overlaps the start of task i, and 0 otherwise:
  - For $j = i$: $(d_i = 0 \wedge r_{ij} = 0) \vee (d_i > 0 \wedge r_{ij} = r_i)$

- For $j \neq i$: $((s_j \leq s_i \wedge e_j > s_i \wedge s_i < e_i) \wedge r_{ij} = r_j) \vee ((s_j > s_i \vee e_j \leq s_i \vee s_i = e_i) \wedge r_{ij} = 0)$

- For each task i (with $i \in [1, n]$) we create a variable sr_i, which is the sum of the resource consumptions of the tasks that overlap the start of task i (task i overlaps its own start), i.e., $sr_i = r_{i1} + \cdots + r_{in}$.

Finally, $(s_1 + d_1 = e_1 \wedge \cdots \wedge s_n + d_n = e_n \wedge sr_1 \leq \mathit{limit} \wedge \cdots \wedge sr_n \leq \mathit{limit}) \Leftrightarrow b$ is the reified constraint. Overall, this reification involves $O(n^2)$ constraints. Alternatively, CUMULATIVE can be reified by a reformulation that uses $O(n \cdot m)$ constraints, where m is the number of time-points of the overall make span. Especially in the context of bin packing problems, n tends to be larger than m.

- Let $\underline{t}$ and $\overline{t}$ be the smallest and largest time-points that any task can cross, respectively.

- For each j in $[\underline{t}, \overline{t}]$, let sr_j be the total resource consumption at time-point j:

$$\sum_{i \in [1,n]} (s_i \leq j < e_i) \cdot r_i = sr_j, \forall j$$

Finally, $(s_1 + d_1 = e_1 \wedge \cdots \wedge s_n + d_n = e_n \wedge sr_{\underline{t}} \leq \mathit{limit} \wedge \cdots \wedge sr_{\overline{t}} \leq \mathit{limit}) \Leftrightarrow b$ is the reified constraint.

CYCLE$(nc, \langle s_1, \ldots, s_n \rangle)$ [6, page 828] holds if $S = \langle s_1, \ldots, s_n \rangle$ is a permutation of $[1, n]$ with nc permutation cycles. It is reified as follows:

- For expressing the PFD part of the reformulation of the implied ALLDIFFERENT(S) constraint, we state the SORT$(S, \langle r_1, \ldots, r_n \rangle)$ PFD constraint.

- The key idea is to extract for each s_i (with $i \in [1, n]$) all the s_j that belong to the same permutation cycle. This is done by stating the following conjunction of $n - 1$ PFD constraints:

$$\text{ELEMENT}(i, S, s_{i,1}) \wedge \text{ELEMENT}(s_{i,1}, S, s_{i,2}) \wedge \cdots \wedge \text{ELEMENT}(s_{i,n-2}, S, s_{i,n-1})$$

- Using the MINIMUM$(\mathit{name}_i, \langle i, s_{i,1}, s_{i,2}, \ldots, s_{i,n-1} \rangle)$ PFD constraint for all $i \in [1, n]$, we determine a unique representative name_i for the permutation cycle containing s_i.

- Using the NVALUE$(nb, \langle \mathit{name}_1, \ldots, \mathit{name}_n \rangle)$ PFD constraint, we determine the number nb of permutation cycles.

Finally, $(r_1 < r_2 \wedge \cdots \wedge r_{n-1} < r_n \wedge nc = nb) \Leftrightarrow b$ is the reified constraint, using the second part of the reformulation of the implied ALLDIFFERENT constraint and a condition on the numbers of permutation cycles.

DIFFN$(\langle \langle o_{11}\ s_{11}\ e_{11}, \ldots, o_{1m}\ s_{1m}\ e_{1m} \rangle, \ldots, \langle o_{n1}\ s_{n1}\ e_{n1}, \ldots, o_{nm}\ s_{nm}\ e_{nm} \rangle \rangle)$ [6, page 872], where o_{ik}, s_{ik}, and e_{ik} (with $i \in [1, n]$ and $k \in [1, m]$) respectively denote the *origin*, *size*, and *end* in dimension k of object i, is reified as follows:

$$\left(\bigwedge_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n \\ i < j}} \bigvee_{1 \leq k \leq m} (s_{ik} = 0 \vee s_{jk} = 0 \vee o_{ik} \geq e_{jk} \vee o_{jk} \geq e_{ik}) \wedge \bigwedge_{\substack{1 \leq i \leq n \\ 1 \leq k \leq m}} (o_{ik} + s_{ik} = e_{ik}) \right) \Leftrightarrow b$$

The constraint holds if each pair of these objects has no overlap. Unlike in all the previous examples, we do not need any PFD constraints here, i.e., $p = 0$.

DISJUNCTIVE$(\langle o_1\ d_1, \ldots, o_n\ d_n \rangle)$ [6, page 912], where o_i and d_i (with $i \in [1, n]$) respectively denote the *origin* and *duration* of task i, is reified by a reformulation that uses the PFD constraints SORT_PERMUTATION [6, page 1778] and ELEMENT for expressing a reordering of the tasks:

$$\text{SORT\_PERMUTATION}(\langle o_1, \ldots, o_n \rangle, \langle p_1, \ldots, p_n \rangle, \langle s_1, \ldots, s_n \rangle) \wedge \bigwedge_{1 \leq i \leq n} \text{ELEMENT}(p_i, \langle d_1, \ldots, d_n \rangle, \mathit{dur}_i)$$

Finally, $(s_1 + \mathit{dur}_1 \leq s_2 \wedge \cdots \wedge s_{n-1} + \mathit{dur}_{n-1} \leq s_n) \Leftrightarrow b$ is the reified constraint. Note that we assume that all durations are strictly positive: this implies that all task origins are distinct, which consequently leads

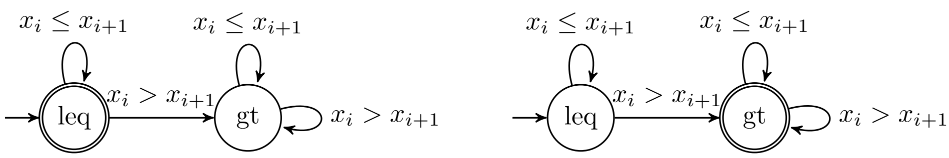

Figure 1: Automaton for the INCREASING constraint, and its complement.

to $\langle p_1, \ldots, p_n \rangle$ being functionally determined by $\langle o_1, \ldots, o_n \rangle$ in the SORT_PERMUTATION($\langle o_1, \ldots, o_n \rangle$, $\langle p_1, \ldots, p_n \rangle, \langle s_1, \ldots, s_n \rangle$) constraint. If some durations can be zero, then one should rather use the reification introduced for the DIFFN constraint.

MINIMUM_WEIGHT_ALLDIFFERENT($\langle v_1, \ldots, v_n \rangle$, *matrix*, *cost*) [6, page 1394], where *matrix* and *cost* respectively denote a matrix providing the cost of assigning value j (with $j \in [1, n]$) to variable v_i (with $i \in [1, n]$) and the sum of the costs of assigning $v_1, \ldots, v_n$, is reified as follows:

$$\begin{gathered}\text{SORT}(\langle v_1, \ldots, v_n \rangle, \langle w_1, \ldots, w_n \rangle) \wedge \\ \text{ELEMENT}(v_1, \mathit{matrix}_1, c_1) \wedge \cdots \wedge \text{ELEMENT}(v_n, \mathit{matrix}_n, c_n) \wedge \\ (w_1 < w_2 \wedge \cdots \wedge w_{n-1} < w_n \wedge c_1 + \cdots + c_n = \mathit{cost}) \Leftrightarrow b\end{gathered}$$

where matrix_i (with $1 \leq i \leq n$) denotes line i of *matrix*, i.e., the costs associated with assigning variable v_i to the values $1, \ldots, n$.

NVALUE($\mathit{nval}, \langle v_1, \ldots, v_n \rangle$) [6, page 1466] is reified as follows:

$$\text{NVALUE}(w, \langle v_1, \ldots, v_n \rangle) \wedge (\mathit{nval} = w) \Leftrightarrow b$$

SORT($\langle v_1, \ldots, v_n \rangle, \langle s_1, \ldots, s_n \rangle$) [6, page 1772] is reified as follows:

$$\text{SORT}(\langle v_1, \ldots, v_n \rangle, \langle t_1, \ldots, t_n \rangle) \wedge (s_1 = t_1 \wedge \cdots \wedge s_n = t_n) \Leftrightarrow b$$

Note that this constraint was only recently added to the set of core global constraints of the Global Constraint Catalogue, for a reason that will become clear in the next section.

4 Categories Used in Reifying Constraints

We now introduce some reification categories that apply to a significant number of constraints of the Global Constraint Catalogue. A constraint may be reifiable according to several categories. Appendix A lists the categories, if any, of each constraint of the Global Constraint Catalogue [6].

In the context of the AUTOMATON meta-constraint [4], a constraint on a sequence X of variables can sometimes be modelled with the help of a finite automaton, possibly with counters, that operates not on X, but on a sequence of *signature variables* that functionally depend via *signature constraints* on a sliding window of variables within X. For example, consider the INCREASING($[x_1, \ldots, x_n]$) constraint, which enforces $x_1 \leq x_2 \leq \cdots \leq x_n$. With the signature constraints $x_i \leq x_{i+1} \Leftrightarrow s_i = 1$ and $x_i > x_{i+1} \Leftrightarrow s_i = 0$, for all $0 \leq i < n$, we get a sequence of $n - 1$ signature variables s_i that can be fed to a finite automaton that recognises the language 1^*. Rather than labelling the transitions of that automaton with *values* of the domain of the signature variables (the set $\{0, 1\}$ in our example), we here label them with the corresponding *conditions* of the signature constraints, as on the left of Figure 1. If each signature variable depends on a sliding window of size j within X (in our example, we have $j = 2$), then we say that the signature constraints are j-ary.

Category $\mathit{Auto}(0, 0)$: Automata without counters and without signature constraints. A constraint that can be modelled by an automaton without counters and without signature constraints (that is via the REGULAR constraint [15]) can be reified with the help of another automaton in the way we already described in [5, page 271], so that there are no PFD constraints in the reformulation ($p = 0$). For example, Figure 2 shows how to reify the GLOBAL_CONTIGUITY [6, page 1058] constraint. There are 19 such constraints in the catalogue.

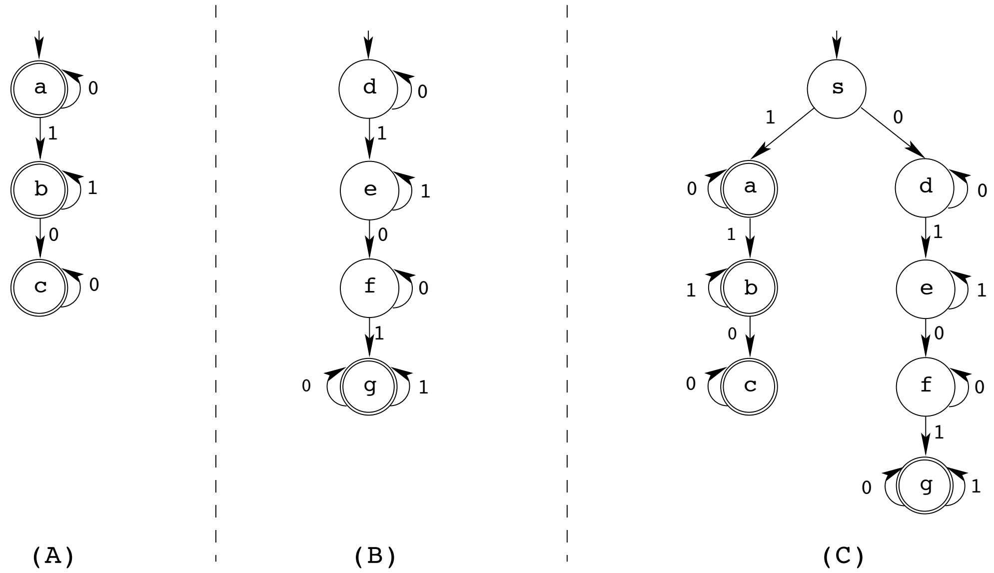

Figure 2: (A) Automaton for recognising solutions to the GLOBAL_CONTIGUITY constraint. (B) Complement automaton, constructed by a standard technique of automata theory, for recognising non-solutions to the GLOBAL_CONTIGUITY constraint. (C) Automaton for the reified GLOBAL_CONTIGUITY constraint, built from the two previous automata upon assuming that the reifying 0-1 variable comes before the sequence of constrained variables.

Category $Auto(0, j > 0)$**: Automata without counters but with signature constraints of arity** $j > 0$**.** A constraint that can be modelled by an automaton without counters but with signature constraints of arity $j > 0$ can be reified as follows: the signature constraints correspond to the PFD constraints, while the automaton itself can be reified as in category $Auto(0, 0)$. For example, consider the automaton on the left of Figure 1 for the INCREASING($[x_1, x_2, \ldots, x_n]$) constraint discussed above. The reification of INCREASING consists of the $n - 1$ signature constraints as the p PFD constraints, together with a constraint for the reified automaton, which is constructed, as in Figure 2(C), from the automaton and its complement, which is on the right of Figure 1. There are 41 such constraints in the catalogue. The 60 constraints of category $Auto(0, j)$ with $j > 0$ or $j = 0$ are annotated in the catalogue with the keyword "reified automaton constraint".

Category $Auto(i > 0, j)$**: Automata with** $i > 0$ **counters.** A constraint that can be modelled by an automaton with $i > 0$ counters $c_1, \ldots, c_i$ with expected values $v_1, \ldots, v_i$ and either without signature constraints ($j = 0$) or with signature constraints σ (of arity $j > 0$) can be reified as follows:

$$\sigma \wedge \text{AUTOMATON}(\ldots) \wedge (w_1 = v_1 \wedge \cdots \wedge w_i = v_i \wedge w_{i+1} = 1) \Leftrightarrow b$$

where:

- c_{i+1} is an auxiliary counter, with initial value 1 if the start state is an accept state, and 0 otherwise,
- $w_1, \ldots, w_{i+1}$ are the counter values in the state where the automaton stops,
- any arc leading to an accept state is amended with the assignment $c_{i+1} \leftarrow 1$,
- any arc leading to a non-accept state is amended with the assignment $c_{i+1} \leftarrow 0$,
- finally, all states are turned into accept states.

Figure 3 shows an example of this transformation. Note that the two previous categories could also be handled like this one. There are 30 such constraints in the catalogue. Except for CHANGE_CONTINUITY, GROUP, and GROUP_SKIP_ISOLATED_ITEM, which fall into the Conj category discussed below, they are the constraints annotated in the catalogue with the keyword "automaton with counters".

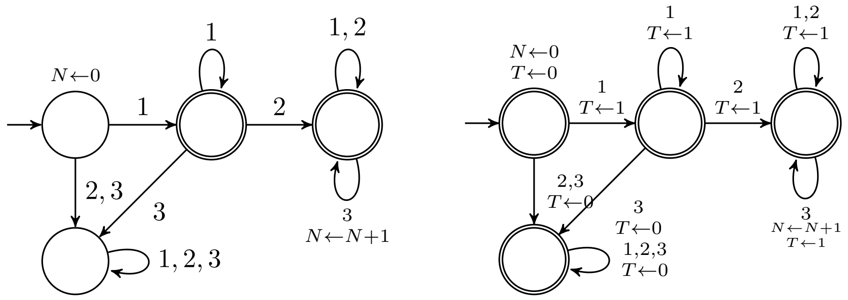

Figure 3: Left: Automaton (without signature constraints) for a constraint over $\{1, 2, 3\}$ requiring that the first 2 be preceded by at least one 1, that the first 3 be preceded by at least one 2, and that there be at least one occurrence of 1; the counter N counts the number of occurrences of 3. Right: Its version used for reification, with an auxiliary counter T, reflecting the truth value.

Category RIC: Built-in reifiable integer constraints. A built-in reifiable integer constraint trivially satisfies our general reification pattern with $p = 0$ PFD constraints. Built-in reifiable constraint correspond to simple arithmetic constraints (e.g., $x \bmod y = 0$) or comparison constraints (e.g., $x \leq y$) involving two integer variables. There are 14 such constraints in the catalogue.

Category RSC: Built-in reifiable set constraints. A built-in reifiable set constraint trivially satisfies our general reification pattern with $p = 0$ PFD constraints. There are 2 such constraints in the catalogue: EQ_SET and IN_SET.

Category Logic: Logical formula involving built-in reifiable constraints. A constraint that can be reformulated as a logical formula involving only built-in reifiable constraints (e.g., ALL_EQUAL, CUMULATIVE, DIFFN). There are 33 such constraints in the catalogue.

Category QLogic: Quantified logical constraints. A category consisting of the geometrical constraints of the catalogue, e.g., DISJOINT_SBOXES. A dedicated sublanguage for encoding such formulas in the context of the GEOST constraint was suggested in [9]. The main purpose of the GEOST constraint is to prevent a collection of geometrical objects from overlapping in multiple dimensions. The non-overlapping condition has a straightforward formulation as a core reifiable constraint. The sublanguage can be seen as syntactic sugar for core reifiable constraints expressing extra conditions, in fact the sublanguage is implemented by unfolding into such constraints considered by GEOST globally together with the non-overlapping condition. There are 17 such constraints in the catalogue, annotated with the keyword "logic".

Category Sort: Constraints with sort in a reformulation. Some constraints involving one or more collections of variables become much simpler to reformulate when these collections are sorted. The SORT constraint can then be used as one of the PFD constraints of the reformulation, and the reified condition involves the sorted variables. Among the core constraints, the ALLDIFFERENT constraint fits this special case, as seen in Section 3. There are 33 such constraints in the catalogue, annotated with the keyword "sort-based reformulation".

Category Conj: Conjunction of reifiable constraints. Constraints that can be reformulated as a conjunction (usually already given in the catalogue) of already reifiable or now reifiable (due to this report) constraints can now be reified. For example, consider the LEX_CHAIN_LESSEQ$([X_1, \ldots, X_n])$ constraint, which requires the sequence of sequences X_i to be lexicographically non-decreasing. It can be reformulated as $\bigwedge_{i \in [1,n-1]}$ LEX_LESSEQ(X_i, X_{i+1}), hence is reifiable as $\bigwedge_{i \in [1,n-1]}$ LEX_LESSEQ$(X_i, X_{i+1}, b_i) \wedge$ $(b_1 = 1 \wedge \cdots \wedge b_{n-1} = 1) \Leftrightarrow b$, since LEX_LESSEQ is now reifiable (as in category $Auto(0, 2)$). There are 21 such constraints in the catalogue.

Category PFD: Pure functional dependency constraints. Following the pattern used in Section 3 for the ELEMENT, GLOBAL_CARDINALITY[_WITH_COSTS], NVALUE, and SORT core constraints, one can reify any PFD constraint. A constraint $c(\mathcal{A}, v_1, \ldots, v_n)$ where the arguments $\mathcal{A}$ functionally determine the variables $v_1, \ldots, v_n$ with no other extra condition is reified into b by $c(\mathcal{A}, w_1, \ldots, w_n) \wedge (v_1 = w_1 \wedge \cdots \wedge v_n = w_n) \Leftrightarrow b$. There are 89 such constraints in the catalogue, annotated with the keyword "pure functional dependency".

Category GenPat: None of the above. A constraint that does not belong to any of the already introduced categories, but that *follows* the general pattern described in Section 2. There are 45 such constraints in the catalogue.

5 Conclusion

Based on the idea that most constraints can naturally be defined by a *determine and test* scheme, where the *determine* part is associated to pure functional dependency (PFD) constraints that determine additional variables, and the *test* part to a core reifiable constraint on these variables, we have shown that most global constraints can be reified. Surprisingly, this simple idea allows us to reify at least 313 of the 381 (i.e., 82%) constraints of the Global Constraint Catalogue. Most of the constraints not covered are graph constraints involving set variables.

Related Work. Some of our insights might be folklore. For instance, Tip 5.3 of [17, page 78] outlines the idea of our PFD category and gives an example, but the notion of PFD and our more general pattern of Section 2 are not identified. Similarly, Example 14 of [13, page 58] also provides an example of negation for a PFD constraint without identifying the pattern. The reformulations of constraints such as ALLDIFFERENT or GLOBAL_CARDINALITY in [8] can be unfolded to make explicit the PFD and core reified constraints. Methods for modifying an automaton for counting string properties were described in [14, page 7] and in [3]. As observed in [11, Section 4], given a global constraint c and its propagator, it is straightforward to construct a propagator for its half-reified version $b \Rightarrow c$, but not so for the only-if version $c \Rightarrow b$. Other work on generic reification does not exploit global constraints, or does not achieve domain consistency, or requires a propagator for the negated constraint: see the discussion in [12]. In the context of software verification, the equivalence of constraint models must sometimes be proven and one needs to negate global constraints [13]. Similarly, the work of [2] is restricted to global constraints that can be reformulated as conjunctions of standard reifiable binary constraints.

Future Work. The obtained reifications can be exploited in many ways. For instance, one can use them for the reformulation of some global constraints. While this may not be very efficient from a memory point of view for a reformulation whose size is quadratic in the number of variables of the constraint, the reformulations within the categories $Auto(0, 0)$, $Auto(0, j > 0)$, $Auto(i > 0, j)$, Sort, Conj, and PFD are quite compact. From a filtering point of view, note that the reformulation of constraints of category PFD is as efficient as the original filtering algorithm of the constraint. Also, one can use these reifications as a simple way for computing the violation cost of constraints. This should be straightforward since the violation cost would be computed only from the part of the reification corresponding to the easily reifiable constraint, and not from the pure-functional-dependency part. It remains to investigate whether these reifications could also be used for generating explanations or for lazy clause generation. Finally, an open question is whether the approach proposed by this report allows us to reify any global constraint. Most likely, in order to address this question formally, one should first provide a formal definition of global constraint that is not linked to any operational aspect.

Acknowledgements

The third and fourth authors are supported by grant 2011-6133 of Vetenskapsrådet (the Swedish Research Council).

References

[1] A. Aggoun and N. Beldiceanu. Extending CHIP in order to solve complex scheduling and placement problems. *Journal of Mathematical Computer Modelling*, 17(7):57–73, 1993.

[2] C. E. Alvarez Divo. Automated reasoning on feature models via constraint programming. Master's thesis, Department of Information Technology, Uppsala University, Sweden, 2011. Available as Report IT 11 041 at `http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-156437`.

[3] N. Beldiceanu, M. Carlsson, P. Flener, and J. Pearson. On matrices, automata and double counting. In A. Lodi, M. Milano, and P. Toth, editors, *Proceedings of CPAIOR'10*, volume 6140 of *LNCS*, pages 10–24. Springer-Verlag, 2010.

[4] N. Beldiceanu, M. Carlsson, and T. Petit. Deriving filtering algorithms from constraint checkers. In M. G. Wallace, editor, *Proceedings of CP'04*, volume 3258 of *LNCS*, pages 107–122. Springer-Verlag, 2004.

[5] N. Beldiceanu, M. Carlsson, and J.-X. Rampon. Global constraint catalog, 2nd Edition. Technical Report T2010:07, Swedish Institute of Computer Science, 2010. Available at `http://soda.swedish-ict.se/view/sicsreport/`.

[6] N. Beldiceanu, M. Carlsson, and J.-X. Rampon. Global constraint catalog, 2nd Edition (revision a). Technical Report T2012:03, Swedish Institute of Computer Science, February 2012. Available at `http://soda.swedish-ict.se/view/sicsreport/`.

[7] N. Beldiceanu and H. Simonis. A constraint seeker: Finding and ranking global constraints from examples. In J. H.-M. Lee, editor, *Proceedings of CP'11*, volume 6876 of *LNCS*. Springer-Verlag, 2011.

[8] C. Bessière, G. Katsirelos, N. Narodytska, C.-G. Quimper, and T. Walsh. Decompositions of all different, global cardinality and related constraints. In C. Boutilier, editor, *Proceedings of IJCAI'09*, pages 419–424, 2009.

[9] M. Carlsson, N. Beldiceanu, and J. Martin. A geometric constraint over k-dimensional objects and shapes subject to business rules. In P. J. Stuckey, editor, *Proceedings of CP'08*, volume 5202 of *LNCS*, pages 220–234. Springer-Verlag, 2008.

[10] COSYTEC. *CHIP Reference Manual*, release 5.1 edition, 1997.

[11] T. Feydy, Z. Somogyi, and P. J. Stuckey. Half reification and flattening. In J. H.-M. Lee, editor, *Proceedings of CP'11*, volume 6876 of *LNCS*, pages 286–301. Springer-Verlag, 2011.

[12] C. Jefferson, N. C. A. Moore, P. Nightingale, and K. E. Petrie. Implementing logical connectives in constraint programming. *Artificial Intelligence*, 174:1407–1429, November 2010.

[13] N. Lazaar. *Méthodologie et outil de test, de localisation de fautes et de correction automatique des programmes à contraintes.* PhD thesis, Rennes 1 University, France, 2011. In French.

[14] J. Menana, S. Demassey, and N. Jussien. Modélisation et optimisation des préférences en planification de personnel. Technical Report Research Report 11-01-INFO, École des Mines de Nantes, 2010. In French.

[15] G. Pesant. A regular language membership constraint for finite sequences of variables. In M. G. Wallace, editor, *Proceedings of CP'04*, volume 3258 of *LNCS*, pages 482–495. Springer-Verlag, 2004.

[16] J.-C. Régin. A filtering algorithm for constraints of difference in CSP. In *Proceedings of AAAI'94*, pages 362–367, 1994.

[17] C. Schulte, G. Tack, and M. Z. Lagerkvist. *Modeling and Programming with Gecode (version 3.7.1)*, October 2011. Available from `http://www.gecode.org/`.

[18] P. Van Hentenryck and Y. Deville. The *cardinality* operator: A new logical connective in constraint logic programming. In *Proceedings of ICLP'91*. MIT Press, 1991.

[19] J. Würtz and T. Müller. Constructive disjunction revisited. In *Proceedings of KI'96*, volume 1137 of *LNAI*, pages 377–386. Springer-Verlag, 1996.

A Classification of Global Constraints w.r.t. Reification

The following table gives for each constraint of the Global Constraint Catalogue [5] the categories, if any, of Section 4 that the constraint belongs to, as well as an optional comment explaining for instance how to reformulate the reified constraint (unless otherwise stated, without considering the restrictions on its arguments). When needed in the third column, the first column gives the template of the constraints, i.e., its name and the structure of its arguments.

Note that, whenever possible, the reformulations given in the table have been done with the intention to be as compact as possible in terms of the size of the reformulation. Whenever there was a choice between a SORT-based reformulation and a GLOBAL_CARDINALITY-based reformulation, we chose the SORT one, as it has the advantage not to make explicit the set of potential values that may be assigned to the variables, since this set may be very large.

An argument is either a scalar (an integer or domain variable) or a collection of tuples of attribute-value pairs. A collection of n tuples with attributes say **a** and **b** is displayed as $\langle a_i, b_i\rangle_{i=1}^{n}$ or as $\langle \mathbf{a}, \mathbf{b}\rangle^n$. A collection displayed without superscript stands for a singleton collection, for example as given in the first argument of ELEM. Nested collections are displayed expanded, e.g., $\left\langle \langle o_{i,j}, s_{i,j}, e_{i,j}\rangle_{j=1}^{m}\right\rangle_{i=1}^{n}$ or $\langle\langle \mathbf{o}, \mathbf{s}, \mathbf{e}\rangle^m\rangle^n$ stands for a collection of n tuples, each consisting of a single attribute whose value is a collection of m tuples with attributes **o**, **s**, and **e**. The same collection notation is used in the reification formulas.

Global Constraint	**Categories**	**Comment**
ABS_VALUE(y, x)	PFD, Logic	$(y = \lvert x\rvert) \Leftrightarrow b$
ALLDIFF_AT_LEAST_K_POS$(k, \langle\langle \mathbf{v}\rangle^m\rangle^n)$	Logic	$\bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} \sum_{\ell=1}^{m} (v_{i,\ell} \neq v_{j,\ell}) \geq k$
ALL_EQUAL$(\langle \mathbf{v}\rangle^n)$	Logic	$(v_1 = v_2 \wedge \cdots \wedge v_{n-1} = v_n) \Leftrightarrow b$
ALL_INCOMPARABLE$(\langle\langle \mathbf{v}\rangle^m\rangle^n)$	Conj	conjunction of INCOMPARABLE constraints on pairs of vectors
ALL_MIN_DIST$(md, \langle \mathbf{v}\rangle^n)$	Sort	SORT$(\langle \mathbf{v}\rangle^n, \langle \mathbf{s}\rangle^n) \wedge (s_2 - s_1 \geq md \wedge \cdots \wedge s_n - s_{n-1} \geq md) \Leftrightarrow b$
ALLDIFFERENT	Sort	see Section 3
ALLDIFFERENT_BETWEEN_SETS	?	set constraint
ALLDIFFERENT_CONSECUTIVE_VALUES$(\langle \mathbf{v}\rangle^n)$	Sort	SORT$(\langle \mathbf{v}\rangle^n, \langle \mathbf{s}\rangle^n) \wedge (s_2 - s_1 = 1 \wedge \cdots \wedge s_n - s_{n-1} = 1) \Leftrightarrow b$
ALLDIFFERENT_CST$(\langle \mathbf{v}, \mathbf{c}\rangle^n)$	Sort	$(u_1 = v_1 + c_1 \wedge \cdots \wedge u_n = v_n + c_n) \wedge$ SORT$(\langle \mathbf{u}\rangle^n, \langle \mathbf{s}\rangle^n) \wedge (s_1 < s_2 \wedge \cdots \wedge s_{n-1} < s_n) \Leftrightarrow b$
ALLDIFFERENT_EXCEPT_0$(\langle \mathbf{v}\rangle^n)$	Sort	SORT$(\langle \mathbf{v}\rangle^n, \langle \mathbf{s}\rangle^n) \wedge ((s_1 = 0 \vee s_1 < s_2) \wedge \cdots \wedge (s_{n-1} = 0 \vee s_{n-1} < s_n)) \Leftrightarrow b$
ALLDIFFERENT_INTERVAL$(\langle \mathbf{v}\rangle^n, si)$	Sort	$(v_1 = si \cdot q_1 + r_1 \wedge \cdots \wedge v_n = si \cdot q_n + r_n) \wedge$ SORT$(\langle \mathbf{q}\rangle^n, \langle \mathbf{s}\rangle^n) \wedge (s_1 < s_2 \wedge \cdots \wedge s_{n-1} < s_n) \Leftrightarrow b$ (with $0 \leq r_i < si$)
ALLDIFFERENT_MODULO$(\langle \mathbf{v}\rangle^n, mod)$	Sort	$(v_1 = mod \cdot p_1 + r_1 \wedge \cdots \wedge v_n = mod \cdot p_n + r_n) \wedge$ SORT$(\langle \mathbf{r}\rangle^n, \langle \mathbf{s}\rangle^n) \wedge (s_1 < s_2 \wedge \cdots \wedge s_{n-1} < s_n) \Leftrightarrow b$ (with $0 \leq r_i < mod$)
ALLDIFF_ON_INTERSECTION$(\langle \mathbf{u}\rangle^m, \langle \mathbf{v}\rangle^n)$	GenPat	Let $\langle \mathbf{w}\rangle^p$ be the values that can be assigned to the variables of $\langle \mathbf{u}\rangle^m$ and $\langle \mathbf{v}\rangle^n$: GCC$(\langle \mathbf{u}\rangle^m, \langle \mathbf{w}, \mathbf{c}\rangle^p) \wedge$ GCC$(\langle \mathbf{v}\rangle^n, \langle \mathbf{w}, \mathbf{d}\rangle^p) \wedge (\bigwedge_{i=1}^{p} (c_i = 1 \wedge d_i = 1) \vee c_i = 0 \vee d_i = 0) \Leftrightarrow b$
ALLDIFFERENT_PARTITION$(\langle \mathbf{v}\rangle^n, \langle\langle \mathbf{val}\rangle^{m_p}\rangle^p)$	Sort	$\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{p} (v_i \in \langle \mathbf{val}_j\rangle^{m_j} \Leftrightarrow u_i = j) \wedge$ SORT$(\langle \mathbf{u}\rangle^n, \langle \mathbf{s}\rangle^n) \wedge (s_1 < s_2 \wedge \cdots \wedge s_{n-1} < s_n) \Leftrightarrow b$

Global Constraint	**Categories**	**Comment**
ALLDIFFERENT_SAME_VALUE$(ns, \langle \mathbf{u} \rangle^n, \langle \mathbf{v} \rangle^n)$	Sort	SORT$(\langle \mathbf{u} \rangle^n, \langle \mathbf{s} \rangle^n) \wedge s = \sum_{i=1}^{n}(u_i = v_i) \wedge (s_1 < s_2 \wedge \cdots \wedge s_{n-1} < s_n \wedge ns = s) \Leftrightarrow b$
ALLPERM$(\langle\langle \mathbf{v} \rangle^m\rangle^n)$	Sort	SORT$(\langle \mathbf{v}_2 \rangle^m, \langle \mathbf{s}_2 \rangle^m) \wedge \cdots \wedge$ SORT$(\langle \mathbf{v}_n \rangle^m, \langle \mathbf{s}_n \rangle^m) \wedge$ LEX_LESSEQ$(\langle \mathbf{v}_1 \rangle^m, \langle \mathbf{s}_2 \rangle^m) \Leftrightarrow b_2 \wedge \cdots \wedge$ LEX_LESSEQ$(\langle \mathbf{v}_1 \rangle^m, \langle \mathbf{s}_n \rangle^m) \Leftrightarrow b_n \wedge (b_2 \wedge \cdots \wedge b_n) \Leftrightarrow b$
AMONG	PFD, Auto(1,1)	
AMONG_DIFF_0	PFD, Auto(1,1)	
AMONG_INTERVAL	PFD, Auto(1,1)	
AMONG_LOW_UP	Auto(1,1)	
AMONG_MODULO	PFD, Auto(1,1)	
AMONG_SEQ$(\ell, u, s, \langle \mathbf{v} \rangle^n, \langle \mathbf{val} \rangle^m)$	Conj	AMONG_LOW_UP$(\ell, u, \langle v_j \rangle_{j=i}^{i+s-1}, \langle \mathbf{val} \rangle^m)$, for all $i \in [1, n-s+1]$
AMONG_VAR	PFD	
AND	PFD, Auto(0,0)	
ARITH	Auto(0,1)	
ARITH_OR	Auto(0,2)	
ARITH_SLIDING	Auto(2,0)	
ASSIGN_AND_COUNTS$(\langle \mathbf{col} \rangle^m, \langle \mathbf{b}, \mathbf{c} \rangle^n, rel, \ell)$	Logic	Let $\langle \mathbf{w} \rangle^p$ be the values that can be assigned to the variables of $\langle \mathbf{b} \rangle^n$:$\bigwedge_{i \in \langle \mathbf{w} \rangle^p}(n_i = \sum_{j=1}^{n}(b_j = i \wedge c_j \in \langle \mathbf{col} \rangle^m)) \wedge (\bigwedge_{i \in \langle \mathbf{w} \rangle^p}(n_i \ rel \ \ell)) \Leftrightarrow b$
ASSIGN_AND_NVALUES$(\langle \mathbf{bin}, \mathbf{val} \rangle^n, rel, \ell)$	GenPat	Let $\langle \mathbf{b} \rangle^p$ be the values that can be assigned to the variables of $\langle \mathbf{bin} \rangle^n$, and let ϵ be an integer not in $\langle \mathbf{b} \rangle^p$: $\bigwedge_{j \in \langle \mathbf{b} \rangle^p} \bigwedge_{i=1}^{n}(bin_i = j \wedge v_{j,i} = val_i) \vee (bin_i \neq j \wedge v_{j,i} = \epsilon) \wedge \bigwedge_{j \in \langle \mathbf{b} \rangle^p}$ NVALUE$(nv_j, \langle \epsilon, v_{j,1}, \ldots, v_{j,n} \rangle) \wedge (\bigwedge_{j \in \langle \mathbf{b} \rangle^p}((nv_j - 1) \ rel \ \ell)) \Leftrightarrow b$
ATLEAST	Auto(1,1)	
ATLEAST_NVALUE$(nval, \langle \mathbf{v} \rangle^n)$	GenPat	NVALUE$(nv, \langle \mathbf{v} \rangle^n) \wedge (nv \geq nval) \Leftrightarrow b$
ATLEAST_NVECTOR$(nvec, \langle\langle \mathbf{v} \rangle^m\rangle^n)$	GenPat	NVECTOR$(nv, \langle\langle \mathbf{v} \rangle^m\rangle^n) \wedge (nv \geq nvec) \Leftrightarrow b$
ATMOST	Auto(1,1)	
ATMOST1	?	set constraint
ATMOST_NVALUE$(nval, \langle \mathbf{v} \rangle^n)$	GenPat	NVALUE$(nv, \langle \mathbf{v} \rangle^n) \wedge (nv \leq nval) \Leftrightarrow b$
ATMOST_NVECTOR$(nvec, \langle\langle \mathbf{v} \rangle^m\rangle^n)$	GenPat	NVECTOR$(nv, \langle\langle \mathbf{v} \rangle^m\rangle^n) \wedge (nv \leq nvec) \Leftrightarrow b$
BALANCE	PFD	
BALANCE_CYCLE	?	
BALANCE_INTERVAL	PFD	
BALANCE_MODULO	PFD	
BALANCE_PARTITION	PFD	
BALANCE_PATH	?	
BALANCE_TREE	?	
BETWEEN_MIN_MAX	Auto(0,2)	
BIN_PACKING	Logic	similar to CUMULATIVE
BIN_PACKING_CAPA	Logic	similar to BIN_PACKING but introduces fixed items wrt. maximum capacity

Global Constraint	**Categories**	**Comment**
BINARY_TREE$(nt, \langle \mathbf{k}, \mathbf{t} \rangle^n)$	GenPat	$\bigwedge_{i=1}^{n}(\text{ELEM}(\langle i, f_{i,1} \rangle, \langle \mathbf{k}, \mathbf{t} \rangle^n) \wedge \bigwedge_{j=1}^{n-1} \text{ELEM}(\langle f_{i,j}, f_{i,j+1} \rangle, \langle \mathbf{k}, \mathbf{t} \rangle^n)) \wedge \text{GCC\_NO\_LOOP}(\ell, \langle \mathbf{t} \rangle^n, \langle i, o_i \rangle_{i=1}^{n}) \wedge \underline{((\bigwedge_{i=1}^{n} f_{i,n-1} = f_{i,n}) \wedge \ell = nt \wedge \bigwedge_{i=1}^{n} o_i \leq 2) \Leftrightarrow b}$ or without using GCC_NO_LOOP, $\bigwedge_{i=1}^{n}(\text{ELEM}(\langle i, f_{i,1} \rangle, \langle \mathbf{k}, \mathbf{t} \rangle^n) \wedge \bigwedge_{j=1}^{n-1} \text{ELEM}(\langle f_{i,j}, f_{i,j+1} \rangle, \langle \mathbf{k}, \mathbf{t} \rangle^n)) \wedge (\bigwedge_{i=1}^{n} f_{i,n-1} = f_{i,n} \wedge nt = \sum_{i=1}^{n}(t_i = i) \wedge \bigwedge_{i=1}^{n} \sum_{j \in [1,n], j \neq i}(t_j = i) \leq 2) \Leftrightarrow b$ (part of the restrictions checked in both reformulations)
BIPARTITE	?	set constraint
CALENDAR	Logic	disjunction of conjunctions of arithmetic constraints
CARDINALITY_ATLEAST	PFD	
CARDINALITY_ATMOST	PFD	
CARDINALITY_ATMOST_PARTITION	PFD	
CHANGE	PFD, Auto(1,2)	
CHANGE_CONTINUITY	Conj	conjunction of constraints of the form Auto(1,2), Auto(1,2), Auto(2,2), Auto(2,2), Auto(2,2), Auto(2,2), Auto(1,2), and Auto(1,2)
CHANGE_PAIR	PFD, Auto(1,4)	
CHANGE_PARTITION	PFD	
CHANGE_VECTORS$(nchange, \langle \langle \mathbf{v} \rangle^m \rangle^n, ctrs)$	PFD, Auto(1,m)	
CIRCUIT$(\langle \mathbf{k}, \mathbf{s} \rangle^n)$	GenPat	when $n > 1$: $\text{ELEM}(\langle 1, t_1 \rangle, \langle \mathbf{k}, \mathbf{s} \rangle^n) \wedge \bigwedge_{i=1}^{n-2} \text{ELEM}(\langle t_i, t_{i+1} \rangle, \langle \mathbf{k}, \mathbf{s} \rangle^n) \wedge \text{SORT}(\langle \mathbf{t} \rangle^{n-1}, \langle \mathbf{r} \rangle^{n-1}) \wedge (1 < r_1 \wedge r_1 < r_2 \wedge \cdots \wedge r_{n-2} < r_{n-1} \wedge r_{n-1} = n) \Leftrightarrow b$ (part of the restrictions checked)
CIRCUIT_CLUSTER	?	
CIRCULAR_CHANGE	PFD, Auto(1,2)	
CLAUSE_AND	Auto(0,0)	
CLAUSE_OR	Auto(0,0)	
CLIQUE	?	set constraint
COLORED_MATRIX	PFD	
COLOURED_CUMULATIVE	GenPat	similar to CUMULATIVE but uses NVALUE instead of SUM_CTR
COLOURED_CUMULATIVES	GenPat	similar to CUMULATIVE but uses NVALUE instead of SUM_CTR
COMMON	PFD	
COMMON_INTERVAL	PFD	
COMMON_MODULO	PFD	
COMMON_PARTITION	PFD	
COMPARE_AND_COUNT$(\langle \mathbf{u} \rangle^n, \langle \mathbf{v} \rangle^n, cp, ct, \ell)$	Logic	$(u_1 \ cp \ v_1 \Leftrightarrow b_1) \wedge \cdots \wedge (u_n \ cp \ v_n \Leftrightarrow b_n) \wedge (b_1 + \cdots + b_n = s) \wedge (s \ ct \ \ell) \Leftrightarrow b$
COND_LEX_COST	Auto(0,0)	
COND_LEX_GREATER	Conj	conjunction of constraints of the form Auto(0,0), Auto(0,0), and $>$
COND_LEX_GREATEREQ	Conj	conjunction of constraints of the form Auto(0,0), Auto(0,0), and $\geq$

Global Constraint	Categories	Comment
COND_LEX_LESS	Conj	conjunction of constraints of the form Auto(0,0), Auto(0,0), and $<$
COND_LEX_LESSEQ	Conj	conjunction of constraints of the form Auto(0,0), Auto(0,0), and $\leq$
CONNECT_POINTS	?	
CONNECTED	?	set constraint
CONSECUTIVE_GROUPS_OF_ONES	Auto(0,0)	
CONSECUTIVE_VALUES($\langle \mathbf{v} \rangle^n$)	Sort	$\textsc{sort}(\langle \mathbf{v} \rangle^n, \langle \mathbf{s} \rangle^n) \wedge (s_2 - s_1 \leq 1 \wedge \cdots \wedge s_{n-1} - s_n \leq 1) \Leftrightarrow b$
CONTAINS_SBOXES	QLogic	
CORRESPONDENCE($\langle \mathbf{f}, \mathbf{p}, \mathbf{t} \rangle^n$)	GenPat	$(\bigwedge_{i=1}^{n} \textsc{element}(p_i, \langle \mathbf{t} \rangle^n, v_i)) \wedge (f_1 = v_1 \wedge \cdots \wedge f_n = v_n) \Leftrightarrow b$
COUNT	Auto(1,1)	
COUNTS	Auto(1,1)	
COVEREDBY_SBOXES	QLogic	
COVERS_SBOXES	QLogic	
CROSSING	PFD	
CUMULATIVE	Logic	see Section 3
CUMULATIVE_CONVEX	?	
CUMULATIVE_PRODUCT	GenPat	similar to CUMULATIVE, but uses PRODUCT_CTR instead of SUM_CTR
CUMULATIVE_TWO_D	?	
CUMULATIVE_WITH_LEVEL_OF_PRIORITY	?	
CUMULATIVES	Logic	similar to CUMULATIVE
CUTSET	?	
CYCLE	GenPat	see Section 3
CYCLE_CARD_ON_PATH	?	
CYCLE_OR_ACCESSIBILITY	?	
CYCLE_RESOURCE	?	
CYCLIC_CHANGE	PFD, Auto(1,2)	
CYCLIC_CHANGE_JOKER	PFD, Auto(1,2)	
DAG	?	set constraint
DECREASING	Auto(0,2)	
DEEPEST_VALLEY	Auto(1,2)	
DERANGEMENT($\langle \mathbf{i}, \mathbf{s} \rangle^n$)	Sort	$\textsc{sort}(\langle \mathbf{s} \rangle^n, \langle \mathbf{t} \rangle^n) \wedge ((s_1 \neq i_1 \wedge \cdots \wedge s_n \neq i_n) \wedge (t_1 < t_2 \wedge \cdots \wedge t_{n-1} < t_n)) \Leftrightarrow b$
DIFFER_FROM_AT_LEAST_K_POS	Auto(1,2)	
DIFFN	Logic	see Section 3
DIFFN_COLUMN($\langle \langle \mathbf{o}, \mathbf{s}, \mathbf{e} \rangle^m \rangle^n, d$)	Logic	$(\bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} \bigvee_{k=1}^{m} (s_{i,k} = 0 \vee s_{j,k} = 0 \vee o_{i,k} \geq e_{j,k} \vee o_{j,k} \geq e_{i,k}) \wedge \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} ((e_{i,d} \leq o_{j,d} \vee e_{j,d} \leq o_{i,d}) \vee (o_{i,d} = o_{j,d} \wedge e_{i,d} = e_{j,d})) \wedge \bigwedge_{i=1}^{n} \bigwedge_{k=1}^{m} (o_{i,k} + s_{i,k} = e_{i,k})) \Leftrightarrow b$
DIFFN_INCLUDE($\langle \langle \mathbf{o}, \mathbf{s}, \mathbf{e} \rangle^m \rangle^n, d$))	Logic	$(\bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} \bigvee_{k=1}^{m} (s_{i,k} = 0 \vee s_{j,k} = 0 \vee o_{i,k} \geq e_{j,k} \vee o_{j,k} \geq e_{i,k}) \wedge \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} ((e_{i,d} \leq o_{j,d} \vee e_{j,d} \leq o_{i,d}) \vee (o_{i,d} \leq o_{j,d} \wedge e_{j,d} \leq e_{i,d}) \vee (o_{j,d} \leq o_{i,d} \wedge e_{i,d} \leq e_{j,d})) \wedge \bigwedge_{i=1}^{n} \bigwedge_{k=1}^{m} (o_{i,k} + s_{i,k} = e_{i,k})) \Leftrightarrow b$
DISCREPANCY	PFD	
DISJ	?	set constraint

Global Constraint	Categories	Comment
DISJOINT$(\langle \mathbf{u} \rangle^m, \langle \mathbf{v} \rangle^n)$	GenPat	Let $\langle \mathbf{w} \rangle^p$ be the values that can be assigned to the variables of $\langle \mathbf{u} \rangle^m$ and $\langle \mathbf{v} \rangle^n$: GLOBAL_CARDINALITY$(\langle \mathbf{u} \rangle^m, \langle \mathbf{w}, \mathbf{c} \rangle^p) \wedge$ GLOBAL_CARDINALITY$(\langle \mathbf{v} \rangle^n, \langle \mathbf{w}, \mathbf{d} \rangle^p) \wedge ((c_1 = 0 \vee d_1 = 0) \wedge \cdots \wedge (c_p = 0 \vee d_p = 0)) \Leftrightarrow b$
DISJOINT_SBOXES	QLogic	
DISJOINT_TASKS$(\langle \mathbf{o}, \mathbf{d}, \mathbf{e} \rangle^m, \langle \mathbf{o}', \mathbf{d}', \mathbf{e}' \rangle^n)$	Logic	$((\bigwedge_{i=1}^m \bigwedge_{j=1}^n (e_i \leq o'_j) \vee (e'_j \leq o_i)) \wedge (\bigwedge_{i=1}^m o_i + d_i = e_i) \wedge (\bigwedge_{i=1}^n o'_i + d'_i = e'_i)) \Leftrightarrow b$
DISJUNCTIVE	Sort	see Section 3
DISJUNCTIVE_OR_SAME_END$(\langle \mathbf{o}, \mathbf{d} \rangle^n)$	Logic	$(\bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^n (d_i = 0 \vee d_j = 0 \vee o_i + d_i \leq o_j \vee o_j + d_j \leq o_i \vee o_i + d_i = o_j + d_j)) \Leftrightarrow b$
DISJUNCTIVE_OR_SAME_START$(\langle \mathbf{o}, \mathbf{d} \rangle^n)$	Logic	$(\bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^n (d_i = 0 \vee d_j = 0 \vee o_i + d_i \leq o_j \vee o_j + d_j \leq o_i \vee o_i = o_j)) \Leftrightarrow b$
DISTANCE	PFD	
DISTANCE_BETWEEN	PFD	
DISTANCE_CHANGE	PFD, Auto(1,4)	
DIVISIBLE(q, d)	RIC	$q = d \cdot k \Leftrightarrow b$
DIVISIBLE_OR(c, d)	Logic	$(c \mod d = 0 \vee d \mod c = 0) \Leftrightarrow b$
DOM_REACHABILITY	?	set constraint
DOMAIN$(\langle \mathbf{v} \rangle^n, \ell, u)$	Logic	$((v_1 \geq \ell \wedge v_1 \leq u) \wedge \cdots \wedge (v_n \geq \ell \wedge v_n \leq u)) \Leftrightarrow b$
DOMAIN_CONSTRAINT	Auto(0,2)	
ELEM$(\langle i, u \rangle, \langle \mathbf{k}, \mathbf{v} \rangle^n)$	PFD, Auto(0,3)	
ELEM_FROM_TO	Auto(0,4)	
ELEMENT	PFD, Auto(0,3)	see Section 3
ELEMENT_GREATEREQ	Auto(0,2)	
ELEMENT_LESSEQ	Auto(0,2)	
ELEMENT_MATRIX	Auto(0,3)	
ELEMENT_PRODUCT$(y, \langle \mathbf{t} \rangle^n, x, z)$	PFD	also GenPat: ELEMENT$(y, \langle \mathbf{t} \rangle^n, v) \wedge u = v \cdot x \wedge z = u \Leftrightarrow b$
ELEMENT_SPARSE	Auto(0,2)	
ELEMENTN$(ind, \langle \mathbf{t} \rangle^n, \langle \mathbf{w} \rangle^m)$	Auto(0,0)	also GenPat: ELEMENT$(ind, \langle \mathbf{t} \rangle^n, v_1) \wedge \cdots \wedge$ ELEMENT$(ind + m - 1, \langle \mathbf{t} \rangle^n, v_m) \wedge (w_1 = v_1 \wedge \cdots \wedge w_m = v_m) \Leftrightarrow b$
ELEMENTS	PFD	
ELEMENTS_ALLDIFFERENT$(\langle \mathbf{i}, \mathbf{v} \rangle^n, \langle \mathbf{ti}, \mathbf{tv} \rangle^n)$	GenPat	$\bigwedge_{k=1}^n$ ELEM$(\langle i_k, v_k \rangle, \langle \mathbf{ti}, \mathbf{tv} \rangle^n) \wedge$ SORT$(\langle \mathbf{i} \rangle^n, \langle \mathbf{s} \rangle^n) \wedge (s_1 < s_2 \wedge \cdots \wedge s_{n-1} < s_n) \Leftrightarrow b$
ELEMENTS_SPARSE$(\langle \mathbf{k}, \mathbf{v} \rangle^n, \langle \mathbf{ind}, \mathbf{val} \rangle^m, d)$	Logic	$(\bigwedge_{i=1}^n (\bigvee_{j=1}^m (k_i = ind_j \wedge v_i = val_j)) \vee (k_i \notin \langle \mathbf{ind} \rangle^m \wedge v_i = d)) \Leftrightarrow b$
EQ	PFD, RIC	
EQ_CST	PFD, RIC	
EQ_SET	RSC	set constraint
EQUAL_SBOXES	QLogic	
EQUIVALENT	PFD, Auto(0,0)	
EXACTLY	PFD, Auto(1,1)	
GCD	PFD	
GEOST	QLogic	
GEOST_TIME	QLogic	
GEQ	RIC	

Global Constraint	Categories	Comment
GEQ_CST	RIC	
GLOBAL_CARDINALITY (GCC)	PFD	see Section 3
GCC_LOW_UP$(\langle\mathbf{x}\rangle^n, \langle\mathbf{v},\mathbf{l},\mathbf{u}\rangle^m)$	GenPat	$\text{GCC}(\langle\mathbf{x}\rangle^n, \langle v_i, o_i\rangle_{i=1}^m) \wedge (\ell_1 \leq o_1 \wedge o_1 \leq u_1 \wedge \cdots \wedge \ell_m \leq o_m \wedge o_m \leq u_m) \Leftrightarrow b$
GCC_LOW_UP_NO_LOOP$(lc, uc, \langle\mathbf{x}\rangle^n, \langle\mathbf{v},\mathbf{l},\mathbf{u}\rangle^m)$	GenPat	$\text{GCC\_NO\_LOOP}(c, \langle\mathbf{x}\rangle^n, \langle v_i, o_i\rangle_{i=1}^m) \wedge (lc \leq c \wedge c \leq uc \wedge \ell_1 \leq o_1 \wedge o_1 \leq u_1 \wedge \cdots \wedge \ell_m \leq o_m \wedge o_m \leq u_m) \Leftrightarrow b$
GCC_NO_LOOP	PFD	
GLOBAL_CARDINALITY_WITH_COSTS	PFD	see Section 3
GLOBAL_CONTIGUITY	Auto(0,0)	see Section 4
GOLOMB$(\langle\mathbf{v}\rangle^n)$	GenPat	$(\bigwedge_{i=2}^n \bigwedge_{j=1}^{i-1} d_{\frac{(i-1)\cdot(i-2)}{2}+j} = v_i - v_j) \wedge \text{SORT}(\langle\mathbf{d}\rangle^m, \langle\mathbf{s}\rangle^m) \wedge (v_1 < v_2 \wedge \cdots \wedge v_{n-1} < v_n \wedge 0 < s_1 \wedge s_1 < s_2 \wedge \cdots \wedge s_{m-1} < s_m) \Leftrightarrow b$, where $m = \frac{(n)\cdot(n-1)}{2}$ and where $v_1 < v_2 \wedge \cdots \wedge v_{n-1} < v_n$ is redundant
GRAPH_CROSSING	PFD	
GRAPH_ISOMORPHISM	?	set constraint
GROUP	Conj	conjunction of constraints of the form Auto(1,0), Auto(2,0), Auto(2,0), Auto(2,0), Auto(2,0), and Auto(1,0)
GROUP_SKIP_ISOLATED_ITEM	Conj	conjunction of constraints of the form Auto(1,0), Auto(2,0), Auto(2,0), and Auto(1,0)
GT	RIC	
HIGHEST_PEAK	Auto(1,2)	
IMPLY	PFD, Auto(0,0)	
IN	Auto(0,1)	
IN_INTERVAL	Auto(0,1)	also reified by IN_INTERVAL_REIFIED
IN_INTERVAL_REIFIED		reifies IN_INTERVAL
IN_INTERVALS$(v, \langle\mathbf{l},\mathbf{u}\rangle^n)$	Logic	$((v \geq \ell_1 \wedge v \leq u_1) \vee \cdots \vee (v \geq \ell_n \wedge v \leq u_n)) \Leftrightarrow b$
IN_RELATION$(\langle\mathbf{v}\rangle^m, \langle\langle\mathbf{t}\rangle^m\rangle^n)$	Logic	$(\vee_{i=1}^n (v_1 = t_{i,1} \wedge \cdots \wedge v_m = t_{i,m})) \Leftrightarrow b$
IN_SAME_PARTITION	Auto(0,2)	
IN_SET	RSC	set constraint
INCOMPARABLE$(\langle\mathbf{u}\rangle^n, \langle\mathbf{v}\rangle^n)$	Sort	$\text{SORT}(\langle\mathbf{u}\rangle^n, \langle\mathbf{r}\rangle^n) \wedge \text{SORT}(\langle\mathbf{v}\rangle^n, \langle\mathbf{s}\rangle^n) \wedge ((r_1 > s_1 \vee \cdots \vee r_n > s_n) \wedge (s_1 > r_1 \vee \cdots \vee s_n > r_n)) \Leftrightarrow b$
INCREASING	Auto(0,2)	see Section 4
INCREASING_GLOBAL_CARDINALITY	Auto(0,0)	
INCREASING_NVALUE	Auto(0,0)	
INCREASING_NVALUE_CHAIN	?	
INCREASING_SUM$(\langle\mathbf{v}\rangle^n, s)$	Conj	conjunction of one single constraint, $\text{SUM\_CTR}(\langle\mathbf{v}\rangle^n, =, s) \Leftrightarrow b$ (note that the fact that $\langle\mathbf{v}\rangle^n$ is increasing is part of the restrictions)
INDEXED_SUM$(\langle\mathbf{k},\mathbf{w}\rangle^m, \langle\mathbf{ind},\mathbf{sum}\rangle^n)$	GenPat	$\bigwedge_{i=1}^n (s_i = \sum_{j=1}^m (ind_i = k_j) \cdot w_j) \wedge (\bigwedge_{i=1}^n sum_i = s_i \wedge \bigwedge_{j=1}^m k_j \in \langle\mathbf{ind}\rangle^n) \Leftrightarrow b$
INFLEXION	Auto(1,2)	
INSIDE_SBOXES	QLogic	
INT_VALUE_PRECEDE	Auto(0,0)	
INT_VALUE_PRECEDE_CHAIN	Auto(0,1)	
INTERVAL_AND_COUNT$(a, \langle\mathbf{c}\rangle^m, \langle\mathbf{o},\mathbf{col}\rangle^n, s)$	Logic	$\bigwedge_{k=1}^{\ell} \bigwedge_{t=1}^n b_{k,t} \Leftrightarrow ((k-1)\cdot s \leq o_t \wedge o_t < k \cdot s \wedge col_t \in \langle\mathbf{c}\rangle^m) \wedge (\bigwedge_{k=1}^{\ell} \sum_{t=1}^n b_{k,t} \leq a) \Leftrightarrow b$, with $\ell = \left\lfloor \frac{\max(\mathbf{o})+s}{s} \right\rfloor$

Global Constraint	Categories	Comment
INTERVAL_AND_SUM$(s, \langle \mathbf{o}, \mathbf{h} \rangle^n, \ell)$	Logic	$\bigwedge_{k=1}^{m} \bigwedge_{t=1}^{n} b_{k,t} \Leftrightarrow ((k-1) \cdot s \leq o_t \wedge o_t < k \cdot s) \wedge (\bigwedge_{k=1}^{m} \sum_{t=1}^{n} b_{k,t} \cdot h_t \leq \ell) \Leftrightarrow b$ with $m = \lfloor \frac{\max(\mathbf{o})+s}{s} \rfloor$
INVERSE$(\langle \mathbf{i}, \mathbf{s}, \mathbf{p} \rangle^n)$	PFD	INVERSE$(\langle \mathbf{i}, \mathbf{s}, \mathbf{q} \rangle^n) \wedge (p_1 = q_1 \wedge \cdots \wedge p_n = q_n) \Leftrightarrow b$
INVERSE_OFFSET	PFD	
INVERSE_SET	?	set constraint
INVERSE_WITHIN_RANGE	?	
ITH_POS_DIFFERENT_FROM_0	Auto(2,1)	
K_ALLDIFFERENT$(\langle \langle \mathbf{v} \rangle^{m_n} \rangle^n)$	Conj	conjunction of n ALLDIFFERENT constraints
K_CUT		set constraint
K_DISJOINT$(\langle \langle \mathbf{v} \rangle^{m_n} \rangle^n)$	GenPat	Let $\langle \mathbf{w} \rangle^p$ be the values that can be assigned to the variables of $\mathbf{v_1}, \ldots, \mathbf{v_n}$: $(\bigwedge_{k=1}^{n}$ GCC$(\langle \mathbf{v_k} \rangle^{m_k}, \langle w_i, c_{i,k} \rangle_{i=1}^{p})) \wedge (\bigwedge_{i=1}^{p} \sum_{k=1}^{n} (c_{i,k} > 0) \leq 1) \Leftrightarrow b$
K_SAME$(\langle \langle \mathbf{v} \rangle^m \rangle^n)$	Sort	$(\bigwedge_{i=1}^{n}$ SORT$(\langle \mathbf{v}_i \rangle^m, \langle \mathbf{s}_i \rangle^m)) \wedge (\bigwedge_{i=1}^{n-1} \bigwedge_{j=1}^{m} s_{i,j} = s_{i+1,j}) \Leftrightarrow b$
K_SAME_INTERVAL	Sort	similar to K_SAME
K_SAME_MODULO	Sort	similar to K_SAME
K_SAME_PARTITION	Sort	similar to K_SAME
K_USED_BY	Sort	similar to K_SAME (introduce unrestricted variables)
K_USED_BY_INTERVAL	Sort	similar to K_SAME (introduce unrestricted variables)
K_USED_BY_MODULO	Sort	similar to K_SAME (introduce unrestricted variables)
K_USED_BY_PARTITION	Sort	similar to K_SAME (introduce unrestricted variables)
LENGTH_FIRST_SEQUENCE	Auto(1,2)	
LENGTH_LAST_SEQUENCE	Auto(1,2)	
LEQ	RIC	
LEQ_CST	RIC	
LEX2	Conj	conjunction of LEX_LESSEQ constraints
LEX_ALLDIFFERENT	Conj	conjunction of LEX_DIFFERENT constraints
LEX_BETWEEN	Auto(0,1)	
LEX_CHAIN_LESS	Conj	conjunction of LEX_LESS constraints
LEX_CHAIN_LESSEQ	Conj	conjunction of LEX_LESSEQ constraints
LEX_DIFFERENT	Auto(0,2)	
LEX_EQUAL	Auto(0,2)	
LEX_GREATER	Auto(0,2)	
LEX_GREATEREQ	Auto(0,2)	
LEX_LESS	Auto(0,2)	
LEX_LESSEQ	Auto(0,2)	
LEX_LESSEQ_ALLPERM$(\langle \mathbf{u} \rangle^n, \langle \mathbf{v} \rangle^n)$	GenPat	SORT$(\langle \mathbf{v} \rangle^n, \langle \mathbf{s} \rangle^n) \wedge (LEX_LESSEQ(\langle \mathbf{u} \rangle^n, \langle \mathbf{s} \rangle^n) \Leftrightarrow b)$
LINK_SET_TO_BOOLEANS	?	set constraint
LONGEST_CHANGE	PFD, Auto(2,2)	
LT	RIC	
MAP	PFD	
MAX_INDEX$(imax, \langle \mathbf{i}, \mathbf{v} \rangle^n)$	GenPat	MAXIMUM$(max, \langle \mathbf{v} \rangle^n) \wedge$ ELEM$(\langle imax, val \rangle, \langle \mathbf{i}, \mathbf{v} \rangle^n) \wedge (max = val \Leftrightarrow b)$
MAX_N	PFD	
MAX_NVALUE	PFD	
MAX_SIZE_SET_OF_CONSECUTIVE_VAR	PFD	

Global Constraint	Categories	Comment
MAXIMUM	PFD, Auto(0,2)	
MAXIMUM_MODULO	PFD	
MEET_SBOXES	QLogic	
MIN_INDEX$(imin, \langle \mathbf{i}, \mathbf{v} \rangle^n)$	GenPat	MINIMUM$(min, \langle \mathbf{v} \rangle^n) \wedge$ ELEM$(\langle imin, val \rangle, \langle \mathbf{i}, \mathbf{v} \rangle^n) \wedge (min = val \Leftrightarrow b)$
MIN_N	PFD	
MIN_NVALUE	PFD	
MIN_SIZE_SET_OF_CONSECUTIVE_VAR	PFD	
MINIMUM	PFD, Auto(0,2)	
MINIMUM_EXCEPT_0	PFD, Auto(0,2)	
MINIMUM_GREATER_THAN	Auto(0,3)	
MINIMUM_MODULO	PFD	
MINIMUM_WEIGHT_ALLDIFFERENT	GenPat	see Section 3
MULTI_GLOBAL_CONTIGUITY	Conj	conjunction of GLOBAL_CONTIGUITY
MULTI_INTER_DISTANCE$(\langle \mathbf{v} \rangle^n, \ell, d)$	Sort	SORT$(\langle \mathbf{v} \rangle^n, \langle \mathbf{s} \rangle^n) \wedge (\bigwedge_{i=1}^{n-\ell} s_i + d \leq s_{i+\ell}) \Leftrightarrow b$
NAND	PFD, Auto(0,0)	
NCLASS	PFD	
NEQ	RIC	
NEQ_CST	RIC	
NEQUIVALENCE	PFD	
NEXT_ELEMENT	Auto(0,4)	
NEXT_GREATER_ELEMENT$(u, v, \langle \mathbf{var} \rangle^n)$	Logic	$(var_1 < var_2 \wedge \ldots \wedge var_{n-1} < var_n \wedge \bigvee_{i \in [1,n]} v = var_i \wedge (i = 1 \vee var_{i-1} \leq u)) \Leftrightarrow b$, note that $u < v$ is part of the restrictions
NINTERVAL	PFD	
NO_PEAK	Auto(0,2)	
NO_VALLEY	Auto(0,2)	
NON_OVERLAP_SBOXES	QLogic	
NOR	PFD, Auto(0,0)	
NOT_ALL_EQUAL	Auto(0,2)	
NOT_IN	Auto(0,1)	
NPAIR	PFD	
NSET_OF_CONSECUTIVE_VALUES	PFD	
NVALUE	PFD	see Section 3
NVALUE_ON_INTERSECTION	PFD	
NVALUES$(\langle \mathbf{v} \rangle^n, relop, \ell)$	GenPat	NVALUE$(nv, \langle \mathbf{v} \rangle^n) \wedge (nv \ relop \ \ell) \Leftrightarrow b$
NVALUES_EXCEPT_0$(\langle \mathbf{v} \rangle^n, relop, \ell)$	GenPat	NVALUE$(nv, \langle \mathbf{v} \rangle^n) \wedge$ AMONG$(z, \langle \mathbf{v} \rangle^n, 0) \wedge (nv - (z > 0) \ relop \ \ell) \Leftrightarrow b$
NVECTOR	PFD	
NVECTORS$(\langle \langle \mathbf{v} \rangle^m \rangle^n, relop, \ell)$	GenPat	NVECTOR$(nv, \langle \langle \mathbf{v} \rangle^m \rangle^n) \wedge (nv \ relop \ \ell) \Leftrightarrow b$
NVISIBLE_FROM_END	PFD	
NVISIBLE_FROM_START	PFD	
OPEN_ALLDIFFERENT	?	set constraint
OPEN_AMONG	?	set constraint
OPEN_ATLEAST	?	set constraint
OPEN_ATMOST	?	set constraint
OPEN_GLOBAL_CARDINALITY	?	set constraint
OPEN_GLOBAL_CARDINALITY_LOW_UP	?	set constraint

Global Constraint	**Categories**	**Comment**
OPEN_MAXIMUM	Auto(0,3)	
OPEN_MINIMUM	Auto(0,3)	
OPPOSITE_SIGN	Logic	$x \cdot y \leq 0 \Leftrightarrow b$
OR	PFD, Auto(0,0)	
ORCHARD	PFD	
ORDERED_ATLEAST_NVECTOR$(a, \langle\langle \mathbf{v}\rangle^m\rangle^n)$	Conj	conjunction of ATLEAST_NVECTOR$(a, \langle\langle \mathbf{v}\rangle^m\rangle^n)$ and LEX_CHAIN_LESSEQ$(\langle\langle \mathbf{v}\rangle^m\rangle^n)$
ORDERED_ATMOST_NVECTOR$(a, \langle\langle \mathbf{v}\rangle^m\rangle^n)$	Conj	conjunction of ATMOST_NVECTOR$(a, \langle\langle \mathbf{v}\rangle^m\rangle^n)$ and LEX_CHAIN_LESSEQ$(\langle\langle \mathbf{v}\rangle^m\rangle^n)$
ORDERED_GCC$(\langle \mathbf{v}\rangle^n, \langle \mathbf{val}, \mathbf{omax}\rangle^m)$	Conj	$\bigwedge_{i=1}^m$ AMONG_LOW_UP$(0, omax_i, \langle \mathbf{v}\rangle^n, \langle val_j\rangle_{j=1}^i)$
ORDERED_NVECTOR$(nv, \langle\langle \mathbf{v}\rangle^m\rangle^n)$	Conj	conjunction of NVECTOR$(nv, \langle\langle \mathbf{v}\rangle^m\rangle^n)$ and LEX_CHAIN_LESSEQ$(\langle\langle \mathbf{v}\rangle^m\rangle^n)$
ORTH_LINK_ORI_SIZ_END$(\langle \mathbf{o}, \mathbf{s}, \mathbf{e}\rangle^n)$	PFD	$(f_1 = o_1 + s_1 \wedge \cdots \wedge f_n = o_n + s_n) \wedge (e_1 = f_1 \wedge \cdots \wedge e_n = f_n) \Leftrightarrow b$
ORTH_ON_THE_GROUND$(\langle \mathbf{o}, \mathbf{s}, \mathbf{e}\rangle^n, vd)$	Logic	$o_{vd} = 1 \Leftrightarrow b$
ORTH_ON_TOP_OF_ORTH	QLogic	
ORTHS_ARE_CONNECTED	?	
OVERLAP_SBOXES	QLogic	
PATH$(np, \langle \mathbf{k}, \mathbf{t}\rangle^n)$	GenPat	$\bigwedge_{i=1}^n($ELEM$(\langle i, f_{i,1}\rangle, \langle \mathbf{k}, \mathbf{t}\rangle^n) \wedge \bigwedge_{j=1}^{n-1}$ ELEM$(\langle f_{i,j}, f_{i,j+1}\rangle, \langle \mathbf{k}, \mathbf{t}\rangle^n)) \wedge$ GCC_NO_LOOP$(\ell, \langle \mathbf{t}\rangle^n, \langle i, o_i\rangle_{i=1}^n) \wedge \underline{((\bigwedge_{i=1}^n f_{i,n-1} = f_{i,n}) \wedge \ell = np \wedge \bigwedge_{i=1}^n o_i \leq 1) \Leftrightarrow b}$ or without using GCC_NO_LOOP, $\bigwedge_{i=1}^n($ELEM$(\langle i, f_{i,1}\rangle, \langle \mathbf{k}, \mathbf{t}\rangle^n) \wedge \bigwedge_{j=1}^{n-1}$ ELEM$(\langle f_{i,j}, f_{i,j+1}\rangle, \langle \mathbf{k}, \mathbf{t}\rangle^n)) \wedge (\bigwedge_{i=1}^n f_{i,n-1} = f_{i,n} \wedge np = \sum_{i=1}^n (t_i = i) \wedge \bigwedge_{i=1}^n \sum_{j \in [1,n], j \neq i} (t_j = i) \leq 1) \Leftrightarrow b$ (part of the restrictions checked in both reformulations)
PATH_FROM_TO	?	set constraint
PATTERN	Auto(0,0)	
PEAK	Auto(1,2)	
PERIOD	PFD	
PERIOD_EXCEPT_0	PFD	
PERIOD_VECTORS	PFD	
PERMUTATION$(\langle \mathbf{v}\rangle^n)$	Sort	SORT$(\langle \mathbf{v}\rangle^n, \langle \mathbf{s}\rangle^n) \wedge (s_1 = 1 \wedge \cdots \wedge s_n = n) \Leftrightarrow b$ (restrictions checked)
PLACE_IN_PYRAMID	QLogic	
POLYOMINO	?	
POWER	PFD	
PRECEDENCE	Logic	$(o_1 + d_1 \leq o_2 \wedge \cdots \wedge o_{n-1} + d_{n-1} \leq o_n) \Leftrightarrow b$
PRODUCT_CTR	RIC	
PROPER_FOREST	?	set constraint
RANGE_CTR	RIC	
RELAXED_SLIDING_SUM$(\ell, m, o, u, s, \langle \mathbf{v}\rangle^n)$	Logic	$(\sum_{i=1}^{n-s+1}(\sum_{j=i}^{i+s-1} v_j \in [o,u]) \in [\ell, m]) \Leftrightarrow b$
REMAINDER	PFD	
ROOTS	?	set constraint
SAME$(\langle \mathbf{u}\rangle^n, \langle \mathbf{v}\rangle^n)$	Sort	SORT$(\langle \mathbf{u}\rangle^n, \langle \mathbf{r}\rangle^n) \wedge$ SORT$(\langle \mathbf{v}\rangle^n, \langle \mathbf{s}\rangle^n) \wedge (r_1 = s_1 \wedge \cdots \wedge r_n = s_n) \Leftrightarrow b$
SAME_AND_GCC$(\langle \mathbf{x}\rangle^n, \langle \mathbf{y}\rangle^n, \langle \mathbf{v}, \mathbf{o}\rangle^m)$	GenPat	GCC$(\langle \mathbf{x}\rangle^n, \langle \mathbf{v}, \mathbf{p}\rangle^m) \wedge$ GCC$(\langle \mathbf{y}\rangle^n, \langle \mathbf{v}, \mathbf{q}\rangle^m) \wedge (o_1 = p_1 \wedge \cdots \wedge o_m = p_m \wedge o_1 = q_1 \wedge \cdots \wedge o_m = q_m) \Leftrightarrow b$

Global Constraint	Categories	Comment
SAME_AND_GCC_LOW_UP$(\langle\mathbf{x}\rangle^n, \langle\mathbf{y}\rangle^n, \langle\mathbf{v},\mathbf{l},\mathbf{u}\rangle^m)$	GenPat	$\text{GCC}(\langle\mathbf{x}\rangle^n, \langle\mathbf{v},\mathbf{p}\rangle^m) \wedge \text{GCC}(\langle\mathbf{y}\rangle^n, \langle\mathbf{v},\mathbf{q}\rangle^m) \wedge (p_1 \in [\ell_1, u_1] \wedge \cdots \wedge p_m \in [\ell_m, u_m] \wedge p_1 = q_1 \wedge \cdots \wedge p_m = q_m) \Leftrightarrow b$
SAME_INTERSECTION$(\langle\mathbf{u}\rangle^m, \langle\mathbf{v}\rangle^n)$	GenPat	Let $\langle\mathbf{w}\rangle^p$ be the values that can be assigned to variables of $\mathbf{u}$ and $\mathbf{v}$: $\text{GCC}(\langle\mathbf{u}\rangle^m, \langle w_i, c_i\rangle_{i=1}^p) \wedge \text{GCC}(\langle\mathbf{v}\rangle^n, \langle w_i, d_i\rangle_{i=1}^p) \wedge (\bigwedge_{i=1}^p c_i = d_i \vee c_i = 0 \vee d_i = 0) \Leftrightarrow b$
SAME_INTERVAL	Sort	similar to SAME
SAME_MODULO	Sort	similar to SAME
SAME_PARTITION	Sort	similar to SAME
SAME_SIGN(v_1, v_2)	Logic	$((v_1 \geq 0 \wedge v_2 \geq 0) \vee (v_1 \leq 0 \wedge v_2 \leq 0)) \Leftrightarrow b$
SCALAR_PRODUCT$(\langle\mathbf{t}\rangle^n, ctr, v)$	GenPat	$\text{SCALAR\_PRODUCT}(\langle\mathbf{t}\rangle^n, =, s) \wedge (s\ ctr\ v) \Leftrightarrow b$
SEQUENCE_FOLDING	Auto(0,2)	
SET_VALUE_PRECEDE	?	set constraint
SHIFT	?	
SIGN_OF	PFD	
SIZE_MAX_SEQ_ALLDIFFERENT	PFD	
SIZE_MAX_STARTING_SEQ_ALLDIFFERENT	PFD	
SLIDING_CARD_SKIP0	Auto(3,3)	
SLIDING_DISTRIBUTION$(s, \langle\mathbf{x}\rangle^n, \langle\mathbf{v},\mathbf{l},\mathbf{u}\rangle^m)$	GenPat	$\bigwedge_{i=1}^{n-s+1} \text{GCC}(\langle x_j\rangle_{j=i}^{i+s-1}, \langle v_j, o_{i,j}\rangle_{j=1}^m) \wedge (\bigwedge_{i=1}^{n-s+1} \bigwedge_{j=1}^m o_{i,j} \in [\ell_j, u_j]) \Leftrightarrow b$
SLIDING_SUM$(\ell, u, s, \langle\mathbf{v}\rangle^n)$	GenPat	$\bigwedge_{i=1}^{n-s+1}(v_i + \cdots + v_{i+s-1} = sum_i) \wedge (\bigwedge_{i=1}^{n-s+1} sum_i \in [\ell, u]) \Leftrightarrow b$
SLIDING_TIME_WINDOW	?	
SLIDING_TIME_WINDOW_FROM_START	?	
SLIDING_TIME_WINDOW_SUM	?	
SMOOTH	PFD, Auto(1,2)	
SOFT_ALL_EQUAL_MAX_VAR$(n, \langle\mathbf{v}\rangle^m)$	GenPat	Let $\langle\mathbf{w}\rangle^p$ be the values that can be assigned to variables or $\mathbf{v}$: $\text{GCC}(\langle\mathbf{v}\rangle^m, \langle w_i, o_i\rangle_{i=1}^p) \wedge \text{MAXIMUM}(max, \langle\mathbf{o}\rangle^p) \wedge (n \leq m - max) \Leftrightarrow b$
SOFT_ALL_EQUAL_MIN_CTR$(n, \langle\mathbf{v}\rangle^m)$	Logic	$(n \leq \sum_{i\in[1,m], j\in[1,m], i\neq j}(v_i = v_j)) \Leftrightarrow b$
SOFT_ALL_EQUAL_MIN_VAR$(n, \langle\mathbf{v}\rangle^m)$	GenPat	Let $\langle\mathbf{w}\rangle^p$ be the values that can be assigned to variables or $\mathbf{v}$: $\text{GCC}(\langle\mathbf{v}\rangle^m, \langle w_i, o_i\rangle_{i=1}^p) \wedge \text{MAXIMUM}(max, \langle\mathbf{o}\rangle^p) \wedge (n \geq m - max) \Leftrightarrow b$
SOFT_ALLDIFFERENT_CTR$(c, \langle\mathbf{v}\rangle^n)$	Logic	$(c \geq \sum_{i=1}^{n-1} \sum_{j=i+1}^n (v_i \neq v_j)) \Leftrightarrow b$
SOFT_ALLDIFFERENT_VAR$(c, \langle\mathbf{v}\rangle^n)$	GenPat	$\text{NVALUE}(m, \langle\mathbf{v}\rangle^n) \wedge (c \geq n - m) \Leftrightarrow b$
SOFT_CUMULATIVE	?	
SOFT_SAME_INTERVAL_VAR	?	
SOFT_SAME_MODULO_VAR	?	
SOFT_SAME_PARTITION_VAR	?	
SOFT_SAME_VAR	?	
SOFT_USED_BY_INTERVAL_VAR	?	
SOFT_USED_BY_MODULO_VAR	?	
SOFT_USED_BY_PARTITION_VAR	?	
SOFT_USED_BY_VAR	?	
SOME_EQUAL$(\langle\mathbf{v}\rangle^n)$	Sort	$\text{SORT}(\langle\mathbf{v}\rangle^n, \langle\mathbf{s}\rangle^n) \wedge (s_1 = s_2 \vee \cdots \vee s_{n-1} = s_n) \Leftrightarrow b$
SORT	PFD	see Section 3
SORT_PERMUTATION$(\langle\mathbf{f}\rangle^n, \langle\mathbf{p}\rangle^n, \langle\mathbf{t}\rangle^n)$	GenPat	$\text{SORT}(\langle\mathbf{f}\rangle^n, \langle\mathbf{sf}\rangle^n) \wedge (\bigwedge_{i=1}^n \text{ELEMENT}(p_i, \langle\mathbf{t}\rangle^n, v_i)) \wedge (sf_1 = t_1 \wedge \cdots \wedge sf_n = t_n \wedge v_1 = f_1 \wedge \cdots \wedge v_n = f_n) \Leftrightarrow b$
STABLE_COMPATIBILITY	?	

Global Constraint	Categories	Comment
STAGE_ELEMENT	PFD, Auto(0,2)	
STRETCH_CIRCUIT	?	
STRETCH_PATH	Auto(0,0)	
STRETCH_PATH_PARTITION	Auto(0,0)	
STRICT_LEX2	Conj	conjunction of LEX_LESS constraints
STRICTLY_DECREASING	Auto(0,2)	
STRICTLY_INCREASING	Auto(0,2)	
STRONGLY_CONNECTED	?	set constraint
SUBGRAPH_ISOMORPHISM	?	set constraint
SUM	?	
SUM_CTR	RIC	
SUM_CUBES_CTR	GenPat	
SUM_FREE	?	set constraint
SUM_OF_INCREMENTS$(\langle \mathbf{v} \rangle^n, \ell)$	Logic	$(v_1 + \sum_{i=2}^{n} \max(v_i - v_{i-1}, 0) \leq \ell) \Leftrightarrow b$
SUM_OF_WEIGHTS_OF_DISTINCT_VALUES	PFD	
SUM_SET	?	set constraint
SUM_SQUARES_CTR	GenPat	
SYMMETRIC	?	set constraint
SYMMETRIC_ALLDIFFERENT$(\langle \mathbf{k}, \mathbf{s} \rangle^n)$	GenPat	SORT$(\langle \mathbf{s} \rangle^n, \langle \mathbf{r} \rangle^n) \wedge \bigwedge_{i=1}^{n}($ELEM$(\langle i, next_i \rangle, \langle \mathbf{k}, \mathbf{s} \rangle^n) \wedge$ ELEM$(\langle next_i, mext_i \rangle, \langle \mathbf{k}, \mathbf{s} \rangle^n)) \wedge (r_1 < r_2 \wedge \cdots \wedge r_{n-1} < r_n \wedge next_1 \neq 1 \wedge \cdots \wedge next_n \neq n \wedge mext_1 = 1 \wedge \cdots \wedge mext_n = n) \Leftrightarrow b$, where SORT$(\langle \mathbf{s} \rangle^n, \langle \mathbf{r} \rangle^n)$ and $r_1 < r_2 \wedge \cdots \wedge r_{n-1} < r_n$ are redundant (part of the restrictions checked)
SYMMETRIC_ALLDIFF_EXCEPT_0$(\langle \mathbf{k}, \mathbf{s} \rangle^n)$	GenPat	SORT$(\langle \mathbf{s} \rangle^n, \langle \mathbf{r} \rangle^n) \wedge \bigwedge_{i=1}^{n}($ELEM$(\langle i, next_i \rangle, \langle \mathbf{k}, \mathbf{s} \rangle^n) \wedge ((next_i = 0 \wedge n_i = i) \vee (next_i \neq 0 \wedge n_i = next_i)) \wedge$ ELEM$(\langle n_i, mext_i \rangle, \langle \mathbf{k}, \mathbf{s} \rangle^n)) \wedge ((r_1 = 0 \vee r_1 < r_2) \wedge \cdots \wedge (r_{n-1} = 0 \vee r_{n-1} < r_n) \wedge next_1 \neq 1 \wedge \cdots \wedge next_n \neq n \wedge ((mext_1 = 0 \wedge next_1 = 0) \vee mext_1 = 1) \wedge \cdots \wedge ((mext_n = 0 \wedge next_n = 0) \vee mext_n = n)) \Leftrightarrow b$ (part of the restrictions checked)
SYMMETRIC_CARDINALITY	?	set constraint
SYMMETRIC_GCC	?	set constraint
TEMPORAL_PATH	?	
TOUR	?	set constraint
TRACK	?	
TREE$(nt, \langle \mathbf{k}, \mathbf{t} \rangle^n)$	GenPat	$\bigwedge_{i=1}^{n}($ELEM$(\langle i, f_{i,1} \rangle, \langle \mathbf{k}, \mathbf{t} \rangle^n) \wedge \bigwedge_{j=1}^{n-1}$ ELEM$(\langle f_{i,j}, f_{i,j+1} \rangle, \langle \mathbf{k}, \mathbf{t} \rangle^n)) \wedge$ GCC_NO_LOOP$(\ell, \langle \mathbf{t} \rangle^n, \langle i, o_i \rangle_{i=1}^{n}) \wedge ((\bigwedge_{i=1}^{n} f_{i,n-1} = f_{i,n}) \wedge \ell = nt) \Leftrightarrow b$ or without using GCC_NO_LOOP, $\bigwedge_{i=1}^{n}($ELEM$(\langle i, f_{i,1} \rangle, \langle \mathbf{k}, \mathbf{t} \rangle^n) \wedge \bigwedge_{j=1}^{n-1}$ ELEM$(\langle f_{i,j}, f_{i,j+1} \rangle, \langle \mathbf{k}, \mathbf{t} \rangle^n)) \wedge (\bigwedge_{i=1}^{n} f_{i,n-1} = f_{i,n} \wedge nt = \sum_{i=1}^{n}(t_i = i)) \Leftrightarrow b$ (part of the restrictions checked in both reformulations)
TREE_RANGE	?	
TREE_RESOURCE	?	

Global Constraint	**Categories**	**Comment**
TWIN($\langle\mathbf{x}\rangle^n, \langle\mathbf{y}\rangle^n$)	GenPat	NVALUE($n_x, \langle\mathbf{x}\rangle^n$) $\wedge$ NVALUE($n_y, \langle\mathbf{y}\rangle^n$) $\wedge$ NVECTOR($n_{xy}, \langle x_i, y_i\rangle_{i=1}^n$) $\wedge$ $(n_x = n_y \wedge n_x = n_{xy} \wedge n_y = n_{xy}) \Leftrightarrow b$, where $n_y = n_{xy}$ is redundant.
TWO_LAYER_EDGE_CROSSING	PFD	
TWO_ORTH_ARE_IN_CONTACT	Auto(0,6) QLogic	
TWO_ORTH_COLUMN	QLogic	
TWO_ORTH_DO_NOT_OVERLAP	Auto(0,6) QLogic	
TWO_ORTH_INCLUDE	QLogic	
USED_BY	Sort	similar to SAME (introduce unrestricted variables)
USED_BY_INTERVAL	Sort	similar to SAME (introduce unrestricted variables)
USED_BY_MODULO	Sort	similar to SAME (introduce unrestricted variables)
USED_BY_PARTITION	Sort	similar to SAME (introduce unrestricted variables)
USES($\langle\mathbf{u}\rangle^m, \langle\mathbf{v}\rangle^n$)	GenPat	Let $\langle\mathbf{w}\rangle^p$ be the values that can be assigned to the variables of $\langle\mathbf{u}\rangle^m$ and $\langle\mathbf{v}\rangle^n$: GCC($\langle\mathbf{u}\rangle^m, \langle w_i, o_i\rangle_{i=1}^p$) $\wedge$ GCC($\langle\mathbf{v}\rangle^n, \langle w_i, q_i\rangle_{i=1}^p$) $\wedge$ $((q_1 = 0 \vee o_1 > 0) \wedge \cdots \wedge (q_p = 0 \vee o_p > 0)) \Leftrightarrow b$
VALLEY	Auto(1,2)	
VEC_EQ_TUPLE($\langle\mathbf{v}\rangle^n, \langle\mathbf{t}\rangle^n$)	Logic	$(v_1 = t_1 \wedge \cdots \wedge v_n = t_n) \Leftrightarrow b$
VISIBLE	?	
WEIGHTED_PARTIAL_ALLDIFF	?	
XOR	PFD, Auto(0,0)	