Rewrite Rules for a Solver for Sets, Binary Relations and Integer Intervals

Maximiliano Cristiá
Universidad Nacional de Rosario

Gianfranco Rossi
Università di Parma

April 22, 2022

Abstract

This document lists in a compact way all the rewrite rules used in the constraint solver for $\mathcal{L}_{\{\cdot\}}$, a constraint language which provides extensional finite sets, binary relations and integer intervals, along with basic operations on them. $\mathcal{L}_{\{\cdot\}}$ is the combination of: $\mathcal{L}_{\mathcal{HFS}}$ (the Boolean algebra of hereditarily finite hybrid sets), $\mathcal{L}_{\mathcal{BR}}$ (a language for finite binary relations encoded as sets of ordered pairs), $\mathcal{L}_{|\cdot|}$ (which extends $\mathcal{L}_{\mathcal{HFS}}$ with the cardinality operator), and $\mathcal{L}_{[]}$ (which extends $\mathcal{L}_{|\cdot|}$ with integer intervals). The constraint solver for $\mathcal{L}_{\{\cdot\}}$ takes the form of a rewrite system acting on $\{\cdot\}$-formulas, i.e., quantifier-free conjunctions and disjunctions of positive and negative $\{\cdot\}$-constraints. A $\{\cdot\}$-*constraint* is any atomic predicate based on a set of predicate symbols Π, respecting the sorts.

Contents

1 Conventions and notation

The set of primitive predicate symbols Π is composed by the following collections of symbols:

- $\{=, \neq\}$ (equality/inequality constraints)
- $\{\in, \mathit{un}, \parallel, \subseteq, \mathit{inters}, \mathit{diff}, \mathit{size}\}$ ((positive) set constraints)
- $\{\mathit{id}, \mathit{inv}, \mathit{comp}, \mathit{dom}, \mathit{ran}, \mathit{dres}, \mathit{dares}, \mathit{rres}, \mathit{rares}, \mathit{apply}, \mathit{rimg}, \mathit{oplus}, \mathit{cp}\}$ ((positive) relational constraints)
- $\{\notin, \mathit{nun}, \nparallel, \mathit{nsubset}, \mathit{ninters}, \mathit{ndiff}, \mathit{nsize}\}$ (negative set constraints)
- $\{\mathit{nid}, \mathit{ninv}, \mathit{ncomp}, \mathit{ndom}, \mathit{nran}, \mathit{ndres}, \mathit{ndares}, \mathit{nrres}, \mathit{nrares}, \mathit{napply}, \mathit{nrimg}, \mathit{noplus}, \mathit{ncp}\}$ (negative relational constraints)
- $\{\mathit{pair}, \mathit{set}, \mathit{rel}, \mathit{pfun}, \mathit{integer}, \mathit{npair}, \mathit{nset}, \mathit{nrel}, \mathit{npfun}, \mathit{ninteger}\}$ (sort constraints).
- $\{\leq, <, >, \geq\}$

We will call *π-constraint* any literal $\pi(x_1, \dots, x_n)$ where π is a symbol in Π.

A *rewrite rule for* π, $\pi \in \Pi$, is a rewrite rule of the form:

$$\phi \rightarrow \Phi$$

where ϕ is a π-constraint and Φ is a $\{\cdot\}$-formula. If Φ has more than one disjunct then the rule is non-deterministic. Conjunctions occurring in Φ have higher precedence than disjunctions.

A *rewriting procedure* for π-constraints consists of the collection of all the rewrite rules for π-constraints. For each rewriting procedure, the solver selects rules in the order they are presented (see figures below). The first rule whose left-hand side matches the input π-constraint c is used to rewrite c. If no rules applies to c, then c is left unchanged (i.e., c is *irreducible*). Note that rule selection is based on pattern-matching, while equality ($=$) in rule bodies represents set unification.

$\mathcal{L}_{\{\cdot\}}$ defines also three sorts Set, Int and O which, intuitively, represent the sort of set, integer and and non-set, non-integer terms, respectively. For notational convenience, the following synonym is also defined: $\mathsf{U} \mathrel{\widehat{=}} \mathsf{O} \cup \mathsf{Set} \cup \mathsf{Int}$.

Notational conventions

- $\mathcal{V}$ denotes a denumerable set of variables partitioned as $\mathcal{V} \mathrel{\widehat{=}} \mathcal{V}_{Set} \cup \mathcal{V}_O \cup \mathcal{V}_{Int}$;
- variable names n and N (possibly with sub and superscripts) are used to denote fresh variables of the proper sort;
- $t_1 \equiv t_2$ (resp., $t_1 \not\equiv t_2$), for any terms t_1 and t_2, means that t_1 is syntactically identical to (resp., distinct from) t_2;
- $\dot{x}$, for any name x, is a shorthand for $x \in \mathcal{V}$;

- $vars(t_1, \ldots, t_n)$ denotes the set of variables occurring in $t_1, \ldots, t_n$.
- $\emptyset$ denotes both the empty set and the interval $[k, m]$ where k and m are constant integer numbers and $m < k$.

2 Sort inference rules

This section lists all the rules applied by procedure sort_infer for inferring sort constraints.

2.1 Sort inference rules for (positive) set constraints

If $t, u, v : \mathsf{U}$ then:

$$t \in u \rightarrow t \in u \wedge set(u) \quad (\text{inf}_1)$$

$$un(t, u, v) \rightarrow un(t, u, v) \wedge set(t) \wedge set(u) \wedge set(v) \quad (\text{inf}_2)$$

$$t \parallel u \rightarrow t \parallel u \wedge set(t) \wedge set(u) \quad (\text{inf}_3)$$

$$t \subseteq u \rightarrow t \subseteq u \wedge set(t) \wedge set(u) \quad (\text{inf}_4)$$

$$inters(t, u, v) \rightarrow inters(t, u, v) \wedge set(t) \wedge set(u) \wedge set(v) \quad (\text{inf}_5)$$

$$diff(t, u, v) \rightarrow diff(t, u, v) \wedge set(t) \wedge set(u) \wedge set(v) \quad (\text{inf}_6)$$

$$size(t, n) \rightarrow size(t, n) \wedge set(t) \wedge integer(n) \quad (\text{inf}_7)$$

Figure 1: Sort inference rules for (positive) set constraints

2.2 Sort inference rules for (positive) relational constraints

If $t, u, v : \mathsf{U}$ then:

$$inv(t,u) \rightarrow inv(t,u) \wedge rel(t) \wedge rel(u) \quad (\text{inf}_8)$$

$$comp(t,u,v) \rightarrow comp(t,u,v) \wedge rel(t) \wedge rel(u) \wedge rel(v) \quad (\text{inf}_9)$$

$$id(t,u) \rightarrow id(t,u) \wedge set(t) \wedge rel(u) \quad (\text{inf}_{10})$$

$$dom(t,u) \rightarrow dom(t,u) \wedge rel(t) \wedge set(u) \quad (\text{inf}_{11})$$

$$ran(t,u) \rightarrow ran(t,u) \wedge rel(t) \wedge set(u) \quad (\text{inf}_{12})$$

$$dres(t,u,v) \rightarrow dres(t,u,v) \wedge set(t) \wedge rel(u) \wedge rel(v) \quad (\text{inf}_{13})$$

$$rres(t,u,v) \rightarrow rres(t,u,v) \wedge rel(t) \wedge set(u) \wedge rel(v) \quad (\text{inf}_{14})$$

$$dares(t,u,v) \rightarrow dares(t,u,v) \wedge set(t) \wedge rel(u) \wedge rel(v) \quad (\text{inf}_{15})$$

$$rares(t,u,v) \rightarrow rares(t,u,v) \wedge rel(t) \wedge set(u) \wedge rel(v) \quad (\text{inf}_{16})$$

$$rimg(t,u,v) \rightarrow rimg(t,u,v) \wedge rel(t) \wedge set(u) \wedge set(v) \quad (\text{inf}_{17})$$

$$oplus(t,u,v) \rightarrow oplus(t,u,v) \wedge rel(t) \wedge rel(u) \wedge rel(v) \quad (\text{inf}_{18})$$

$$apply(t,u,v) \rightarrow apply(t,u,v) \wedge rel(t) \quad (\text{inf}_{19})$$

$$cp(t,u,v) \rightarrow cp(t,u,v) \wedge set(t) \wedge set(u) \wedge rel(v) \quad (\text{inf}_{20})$$

$$rel(t) \rightarrow rel(t) \wedge set(t) \quad (\text{inf}_{21})$$

$$pfun(t) \rightarrow pfun(t) \wedge rel(t) \quad (\text{inf}_{22})$$

Figure 2: Sort inference rules for (positive) relational constraints

2.3 Sort inference rules for negative constraints

The sort inference rules for negative constraints are basically the same used for the positive case, but each predicate name is replaced by its corresponding negative counterpart.

2.4 Sort inference rules for integer constraints

The sort inference rules for integer constraints conjoin an *integer* constraint for each of the involved arguments. For example, $a \leq b \rightarrow a \leq b \wedge integer(a) \wedge integer(b)$.

2.5 Sort inference rules for set terms

In addition, the function find_set is used to find set terms, possibly occurring inside other terms, and to generate the corresponding *set* constraints. The definition of find_set is shown in Figure 3. We assume that all the *true* constraints possibly generated by find_set are immediately removed via a trivial pre-processing.

$$
\begin{array}{l}
\mathsf{find\_set}(t): \\
\quad \text{if } t \equiv X \text{ or } t \text{ is a constant symbol then return } \mathit{true}; \\
\quad \text{if } t \equiv f(t_1, \dots, t_n),\ n > 0, \text{ and } f \not\equiv \{\cdot \mid \cdot\} \\
\qquad \text{then return } \mathsf{find\_set}(t_1) \wedge \cdots \wedge \mathsf{find\_set}(t_n); \\
\quad \text{if } t \equiv \{t_1, \dots, t_n \mid t\} \\
\qquad \text{then return } \mathsf{find\_set}(t_1) \wedge \cdots \wedge \mathsf{find\_set}(t_n) \wedge \mathit{set}(t);
\end{array}
\tag{2.1}
$$

Figure 3: Finding set terms

Remark 1 $\mathcal{L}_{\{\cdot\}}$ *does not provide any sort declarations. Hence, literals in the input formula may be ill-sorted (e.g.* $x \in 1$*). All ill-sorted literals are detected by the solver at run-time and cause the input constraint to be rewritten to false. Ill-sorted literals are detected either by some rewrite rule (e.g.,* $un(1, 2, \emptyset)$ *is rewritten to false thanks to rule* $(\cup_2)$*), or by sort constraints.*

Sort constraints are added to the input formula either by the user or automatically by the solver through the procedure sort_infer *which applies the rules shown in this section. For example, if the input formula is* $x \in 1$ *then it is rewritten to* $x \in 1 \wedge set(1)$ *by rule* (inf_1)*; in the further processing of this formula, literal* $x \in 1$ *is found to be irreducible since no rewrite rule for* $\in$*-constraints applies to it (see Fig. 7), while literal* $set(1)$ *is rewritten to false by the rewrite rules for set-constraints (see Fig. 26); hence, the whole formula is rewritten to false.*

3 Rewrite rules for equality constraints

3.1 Equality

Syntax: $t_1 = t_2$.
Informal semantics: t_1 and t_2 are equal.
Rewrite rules: see Fig. 4-5.

If $x, y, t, x_i, y_i : \mathsf{U}$; $A, B : \mathsf{Set}$; $k, m, i, j : \mathsf{Int}$ then:

$$\dot{x} = \dot{x} \to true \qquad (=_1)$$

If $t \notin \mathcal{V}$:$t = \dot{x} \to \dot{x} = t \qquad (=_2)$

If $\dot{A} \notin vars(x_1, \ldots, x_n)$: $\dot{A} = \{x_1, \ldots, x_n \sqcup \dot{A}\} \to \dot{A} = \{x_1, \ldots, x_n \sqcup N\} \qquad (=_3)$

If $\dot{x} \in vars(t)$: $\dot{x} = t \to false \qquad (=_4)$

If $\dot{x}$ occurs in other literals of the input formula:
$\dot{x} = t \to \dot{x} = t$and substitute $\dot{x}$ by t in all other literals $\qquad (=_5)$

$$\emptyset = [k, m] \to [k, m] = \emptyset \qquad (=_6)$$

$$[k, m] = \emptyset \to m < k \qquad (=_7)$$

$$\{x \sqcup A\} = [k, m] \to [k, m] = \{x \sqcup A\} \qquad (=_8)$$

$$[k, m] = \{x \sqcup A\} \to \{x \sqcup A\} \subseteq [k, m] \wedge size(\{x \sqcup A\}, m - k + 1) \qquad (=_9)$$

If $f \not\equiv g$: $f(x_1, \ldots, x_n) = g(y_1, \ldots, y_m) \to false \qquad (=_{10})$

Figure 4: Rewrite rules for =-constraints (first)

Irreducible form:

- $\dot{x} = t$ and neither t nor the other literals of the formula contain $\dot{x}$.

If $x, y, x_i, y_i : \mathsf{U}$; $A, B : \mathsf{Set}$; $k, m, i, j : \mathsf{Int}$ then:

$$
\begin{aligned}
&\text{If } j \in 1..n{:}\{x_1, \ldots, x_m \sqcup \dot{A}\} = \{y_1, \ldots, y_n \sqcup \dot{A}\} \rightarrow \\
&\quad x_1 = y_j \wedge \{x_2, \ldots, x_m \sqcup \dot{A}\} = \{y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_n \sqcup \dot{A}\} \\
&\quad \vee\ x_1 = y_j \wedge \{x_1, \ldots, x_m \sqcup \dot{A}\} = \{y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_n \sqcup \dot{A}\} \\
&\quad \vee\ x_1 = y_j \wedge \{x_2, \ldots, x_m \sqcup \dot{A}\} = \{y_1, \ldots, y_n \sqcup \dot{A}\} \\
&\quad \vee\ \dot{A} = \{x_1 \sqcup N\} \wedge \{x_2, \ldots, x_m \sqcup N\} = \{y_1, \ldots, y_n \sqcup N\}
\end{aligned} \tag{$=_{11}$}
$$

$$
\begin{aligned}
&\{x \sqcup A\} = \{y \sqcup B\} \rightarrow \\
&\quad x = y \wedge A = B \\
&\quad \vee\ x = y \wedge \{x \sqcup A\} = B \\
&\quad \vee\ x = y \wedge A = \{y \sqcup B\} \\
&\quad \vee\ A = \{y \sqcup N\} \wedge \{x \sqcup N\} = B
\end{aligned} \tag{$=_{12}$}
$$

$$[k, m] = [i, j] \rightarrow (k \leq m \wedge i \leq j \wedge k = i \wedge m = j) \vee (m < k \wedge j < i) \tag{$=_{13}$}$$

$$f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \rightarrow x_1 = y_1 \wedge \cdots \wedge x_n = y_n \tag{$=_{14}$}$$

Figure 5: Rewrite rules for =-constraints (second)

3.2 Inequality

Syntax: $t_1 \neq t_2$.
Informal semantics: t_1 is different from t_2.
Rewrite rules: see Fig. 6.

If $x, y, t, x_i, y_i : \mathsf{U}$; $A, B : \mathsf{Set}$; $k, m, i, k : \mathsf{Int}$ then:

$$\dot{x} \neq \dot{x} \to \mathit{false} \tag{$\neq_1$}$$

$$\text{If } t \notin \mathcal{V}\text{: } t \neq \dot{x} \to \dot{x} \neq t \tag{$\neq_2$}$$

$$\begin{aligned}&\text{If } \dot{A} \notin \mathit{vars}(x_1, \dots, x_n) : \\ &\quad \dot{A} \neq \{x_1, \dots, x_n \sqcup \dot{A}\} \to x_1 \notin \dot{A} \vee \dots \vee x_n \notin \dot{A}\end{aligned} \tag{$\neq_3$}$$

$$\text{If } \dot{x} \in \mathit{vars}(t)\text{:} \dot{x} \neq t \to \mathit{true} \tag{$\neq_4$}$$

$$\begin{aligned}&\{x \sqcup A\} \neq \{y \sqcup B\} \to \\ &\quad (N \in \{x \sqcup A\} \wedge N \notin \{y \sqcup B\}) \vee (N \notin \{x \sqcup A\} \wedge N \in \{y \sqcup B\})\end{aligned} \tag{$\neq_5$}$$

$$\begin{aligned}&[k, m] \neq [i, j] \to \\ &\quad (k \leq m \wedge (m \neq j \vee j < i \vee k \neq i)) \vee (i \leq j \wedge (m \neq j \vee m < k \vee k \neq i))\end{aligned} \tag{$\neq_6$}$$

$$f \neq f \to \mathit{false} \tag{$\neq_7$}$$

$$f(x_1, \dots, x_n) \neq f(y_1, \dots, y_n) \to x_1 \neq y_1 \vee \dots \vee x_n \neq y_n \tag{$\neq_8$}$$

$$\text{If } m \not\equiv n\text{: } f(x_1, \dots, x_m) \neq g(y_1, \dots, y_n) \to \mathit{true} \tag{$\neq_9$}$$

$$\emptyset \neq [k, m] \to [k, m] \neq \emptyset \tag{$\neq_{10}$}$$

$$[k, m] \neq \emptyset \to k \leq m \tag{$\neq_{11}$}$$

$$\{x \sqcup A\} \neq [k, m] \to [k, m] \neq \{x \sqcup A\} \tag{$\neq_{12}$}$$

$$\begin{aligned}&[k, m] \neq \{y \sqcup B\} \to \\ &\qquad (N \in [k, m] \wedge N \notin \{y \sqcup B\}) \vee (N \notin [k, m] \wedge N \in \{y \sqcup B\})\end{aligned} \tag{$\neq_{13}$}$$

$$\text{If } f \not\equiv g\text{: } f(x_1, \dots, x_m) \neq g(y_1, \dots, y_n) \to \mathit{true} \tag{$\neq_{14}$}$$

Figure 6: Rewrite rules for $\neq$-constraints

Irreducible form:

- $\dot{x} \neq t$ and $\dot{x}$ does not occur neither in t nor as an argument of any predicate $p(\dots)$, $p \in \{\mathit{un}, \subseteq, \mathit{inters}, \mathit{dom}, \mathit{ran}, \mathit{id}, \mathit{inv}, \mathit{comp}, \mathit{size}\}$, in the input formula.

4 Rewrite rules for (positive) set constraints

4.1 Membership

Syntax: $t_1 \in t_2$.
Informal semantics: if t_2 is a set, then t_1 is a member of t_2.
Rewrite rules: see Fig. 7.

If $x, y : \mathsf{U}$; $A : \mathsf{Set}$; $k, m : \mathsf{Int}$ then:

$$x \in \emptyset \rightarrow \mathit{false} \quad (\in_1)$$

$$x \in \{y \sqcup A\} \rightarrow x = y \vee x \in A \quad (\in_2)$$

$$x \in \dot{A} \rightarrow \dot{A} = \{x \sqcup N\} \quad (\in_3)$$

$$x \in [k, m] \rightarrow k \leq x \leq m \quad (\in_4)$$

Figure 7: Rewrite rules for $\in$-constraints

Irreducible form: none.

4.2 Union

Syntax: $un(t_1, t_2, t_3)$.
Informal semantics: if t_1, t_2 and t_3 are sets, then $t_3 = t_1 \cup t_2$.
Rewrite rules: see Fig. 8-10.

If $x : \mathsf{U}$; $A, B, C : \mathsf{Set}$ then:

$$un(\dot{A}, \dot{A}, B) \rightarrow \dot{A} = B \qquad (\cup_1)$$

$$un(A, B, \emptyset) \rightarrow A = \emptyset \wedge B = \emptyset \qquad (\cup_2)$$

$$un(\emptyset, A, \dot{B}) \rightarrow \dot{B} = A \qquad (\cup_3)$$

$$un(A, \emptyset, \dot{B}) \rightarrow \dot{B} = A \qquad (\cup_4)$$

If $A \not\equiv [\cdot, \cdot]$:

$$\begin{aligned} & un(\{x \sqcup C\}, A, \dot{B}) \rightarrow \\ & \quad (x \notin A \wedge un(N_1, A, N) \vee A = \{x \sqcup N_2\} \wedge un(N_1, N_2, N)) \\ & \quad \wedge \{x \sqcup C\} = \{x \sqcup N_1\} \wedge \dot{B} = \{x \sqcup N\} \end{aligned} \qquad (\cup_5)$$

If $A \not\equiv [\cdot, \cdot]$:

$$\begin{aligned} & un(A, \{x \sqcup C\}, \dot{B}) \rightarrow \\ & \quad (x \notin A \wedge un(N_1, A, N) \vee A = \{x \sqcup N_2\} \wedge un(N_1, N_2, N)) \\ & \quad \wedge \{x \sqcup C\} = \{x \sqcup N_1\} \wedge \dot{B} = \{x \sqcup N\} \end{aligned} \qquad (\cup_6)$$

If $A \not\equiv [\cdot, \cdot]$ and $B \not\equiv [\cdot, \cdot]$:

$$\begin{aligned} & un(A, B, \{x \sqcup C\}) \rightarrow \\ & \quad (A = \{x \sqcup N_1\} \wedge un(N_1, B_2, N) \\ & \qquad \vee B = \{x \sqcup N_1\} \wedge un(A, N_1, N) \\ & \qquad \vee A = \{x \sqcup N_1\} \wedge B = \{x \sqcup N_2\} \wedge un(N_1, N_2, N)) \\ & \quad \wedge \{x \sqcup C\} = \{x \sqcup N\} \end{aligned} \qquad (\cup_7)$$

Figure 8: Rewrite rules for *un*-constraints (first)

If $A, B : \mathsf{Set}$; $k, m, i, j : \mathsf{Int}$; $A \not\equiv [\cdot,\cdot]$ and $B \not\equiv [\cdot,\cdot]$ then:

$$
\begin{aligned}
& un([k,m], A, B) \rightarrow && (\cup_8)\\
& \quad m < k \wedge A = B \\
& \quad \vee\ k \leq m \wedge \dot{N} \subseteq [k,m] \wedge size(\dot{N}, m-k+1) \wedge un(\dot{N}, A, B)
\end{aligned}
$$

$$
\begin{aligned}
& un(A, [k,m], B) \rightarrow && (\cup_9)\\
& \quad m < k \wedge A = B \\
& \quad \vee\ k \leq m \wedge \dot{N} \subseteq [k,m] \wedge size(\dot{N}, m-k+1) \wedge un(A, \dot{N}, B)
\end{aligned}
$$

$$
\begin{aligned}
& un(A, B, [k,m]) \rightarrow && (\cup_{10})\\
& \quad m < k \wedge A = \emptyset \wedge B = \emptyset \\
& \quad \vee\ (k \leq m \wedge \dot{N} \subseteq [k,m] \wedge size(\dot{N}, m-k+1) \wedge un(A, B, \dot{N}))
\end{aligned}
$$

$$
\begin{aligned}
& un([k,m], [i,j], A) \rightarrow && (\cup_{11})\\
& \quad (m < k \wedge j < i \wedge A = \emptyset) \\
& \quad \vee\ (m < k \wedge i \leq j \wedge [i,j] = A) \\
& \quad \vee\ (k \leq m \wedge j < i \wedge [k,m] = A) \\
& \quad \vee\ (k \leq m \wedge i \leq j \\
& \qquad \wedge\ \dot{N}_1 \subseteq [k,m] \wedge size(\dot{N}_1, m-k+1) \\
& \qquad \wedge\ \dot{N}_2 \subseteq [i,j] \wedge size(\dot{N}_2, j-i+1) \\
& \qquad \wedge\ un(\dot{N}_1, \dot{N}_2, A))
\end{aligned}
$$

$$
\begin{aligned}
& un([k,m], A, [i,j]) \rightarrow && (\cup_{12})\\
& \quad j < i \wedge [k,m] = A = \emptyset \\
& \quad \vee\ i \leq j \wedge m < k \wedge A = [i,j] \\
& \quad \vee\ k \leq m \wedge i \leq j \\
& \qquad \wedge\ \dot{N}_1 \subseteq [k,m] \wedge size(\dot{N}_1, m-k+1) \\
& \qquad \wedge\ \dot{N}_2 \subseteq [i,j] \wedge size(\dot{N}_2, j-i+1) \\
& \qquad \wedge\ un(\dot{N}_1, A, \dot{N}_2)
\end{aligned}
$$

Figure 9: Rewrite rules for *un*-constraints (second)

If A : Set; k, m, i, j, p, q : Int; $A \not\equiv [\cdot, \cdot]$ then:

$$
\begin{aligned}
& un(A, [k,m], [i,j]) \rightarrow && (\cup_{13}) \\
& \quad j < i \wedge [k,m] = A = \emptyset \\
& \quad \vee\ i \leq j \wedge m < k \wedge A = [i,j] \\
& \quad \vee\ k \leq m \wedge i \leq j \\
& \qquad \wedge\ \dot{N}_1 \subseteq [k,m] \wedge size(\dot{N}_1, m - k + 1) \\
& \qquad \wedge\ \dot{N}_2 \subseteq [i,j] \wedge size(\dot{N}_2, j - i + 1) \\
& \qquad \wedge\ un(A, \dot{N}_1, \dot{N}_2)
\end{aligned}
$$

$$
\begin{aligned}
& un([k,m], [i,j], [p,q]) \rightarrow && (\cup_{14}) \\
& \quad (m < k \wedge [i,j] = [p,q]) \\
& \quad \vee\ (j < i \wedge [k,m] = [p,q]) \\
& \quad \vee\ (k \leq m \wedge i \leq j \wedge k \leq i \wedge i \leq m + 1 \wedge m \leq j \wedge p = k \wedge q = j) \\
& \quad \vee\ (k \leq m \wedge i \leq j \wedge k \leq i \wedge i \leq m + 1 \wedge j < m \wedge p = k \wedge q = m) \\
& \quad \vee\ (k \leq m \wedge i \leq j \wedge i < k \wedge k \leq j + 1 \wedge m \leq j \wedge p = i \wedge q = j) \\
& \quad \vee\ (k \leq m \wedge i \leq j \wedge i < k \wedge k \leq j + 1 \wedge j < m \wedge p = i \wedge q = m)
\end{aligned}
$$

Figure 10: Rewrite rules for *un*-constraints (third)

Irreducible form:

- $un(\dot{A}, \dot{B}, \dot{C})$, $\dot{A}$ and $\dot{B}$ distinct variables.

4.3 Disjointness

Syntax: $t_1 \parallel t_2$.
Informal semantics: if t_1 and t_2 are sets, then $t_1 \cap t_2 = \emptyset$.
Rewrite rules: see Fig. 11.

If $x, y : \mathsf{U}$; $A, B : \mathsf{Set}$; $k, m, i, j : \mathsf{Int}$ then:

$$\emptyset \parallel A \to \mathit{true} \tag{$\parallel_1$}$$

$$A \parallel \emptyset \to \mathit{true} \tag{$\parallel_2$}$$

$$\dot{A} \parallel \dot{A} \to \dot{A} = \emptyset \tag{$\parallel_3$}$$

$$\{x \sqcup B\} \parallel \dot{A} \to x \notin \dot{A} \wedge \dot{A} \parallel B \tag{$\parallel_4$}$$

$$\dot{A} \parallel \{x \sqcup B\} \to x \notin \dot{A} \wedge \dot{A} \parallel B \tag{$\parallel_5$}$$

$$\{x \sqcup A\} \parallel \{y \sqcup B\} \to x \neq y \wedge x \notin B \wedge y \notin A \wedge A \parallel B \tag{$\parallel_6$}$$

$$[k, m] \parallel [i, j] \to m < k \vee j < i \vee (k \leq m \wedge i \leq j \wedge (m < i \vee j < k)) \tag{$\parallel_7$}$$

$$A \parallel [k, m] \to [k, m] \parallel A \tag{$\parallel_8$}$$

$$\begin{aligned} &[k, m] \parallel A \to \\ &\quad m < k \vee (k \leq m \wedge \dot{N} \subseteq [k, m] \wedge \mathit{size}(\dot{N}, m - k + 1) \wedge \dot{N} \parallel A) \end{aligned} \tag{$\parallel_9$}$$

Figure 11: Rewrite rules for $\parallel$-constraints (disjointness)

Irreducible form:

- $\dot{A} \parallel \dot{B}$, $\dot{A}$ and $\dot{B}$ distinct variables.

4.4 Size (set cardinality)

Syntax: $size(t_1, t_2)$.
Informal semantics: if t_1 is a set and t_2 is an integer term, then t_2 is the cardinality of t_1.
Rewrite rules: see Fig. 12.

If $x : \mathsf{U}$; $A : \mathsf{Set}$; $k, m, p : \mathsf{Int}$ then:

$$size(\emptyset, m) \to m = 0 \tag{4.1}$$

$$size(A, 0) \to A = \emptyset \tag{4.2}$$

If e is a compound arithmetic expression:

$$size(A, e) \to size(A, \dot{n}) \wedge \dot{n} = e \wedge 0 \leq \dot{n} \tag{4.3}$$

$$\begin{aligned} & size(\{x \sqcup A\}, m) \to \\ & \quad x \notin A \wedge m = 1 + \dot{n} \wedge size(A, \dot{n}) \wedge 0 \leq \dot{n} \\ & \quad \vee A = \{x \sqcup \dot{N}\} \wedge x \notin \dot{N} \wedge size(\dot{N}, m) \end{aligned} \tag{4.4}$$

$$size([k, m], p) \to (m < k \wedge p = 0) \vee (k \leq m \wedge p = m - k + 1) \tag{4.5}$$

Figure 12: Rewrite rules for *size*-constraints

Irreducible forms:

- $size(\dot{A}, c)$, c constant integer number, $c \not\equiv 0$

4.5 Subset

Syntax: $t_1 \subseteq t_2$.
Informal semantics: if t_1 and t_2 are sets, then t_1 is a subset of t_2.
Rewrite rules: see Fig. 13.

If $x, x_i, y, y_i : \mathsf{U}$; $A, B : \mathsf{Set}$; $k, m : \mathsf{Int}$ then:

$$\dot{A} \subseteq \dot{A} \to \mathit{true} \tag{4.6}$$

$$\emptyset \subseteq A \to \mathit{true} \tag{4.7}$$

$$\dot{A} \subseteq \emptyset \to \dot{A} = \emptyset \tag{4.8}$$

$$\{x \sqcup A\} \subseteq \emptyset \to \mathit{false} \tag{4.9}$$

$$\{x_1, \dots, x_n \sqcup \dot{A}\} \subseteq \dot{A} \to \mathit{un}(\{x_1, \dots, x_n \sqcup \dot{A}\}, \dot{A}, \dot{A}) \tag{4.10}$$

$$\begin{aligned} &\{x \sqcup A\} \subseteq \dot{B} \to \\ &\qquad \dot{B} = \{x \sqcup N\} \wedge A \subseteq \{x \sqcup N\} \end{aligned} \tag{4.11}$$

$$\begin{aligned} &\{x_1, \dots, x_n \sqcup \dot{A}\} \subseteq \{y_1, \dots, y_m \sqcup \dot{A}\} \to \\ &\quad \mathit{un}(\{x_1, \dots, x_n \sqcup \dot{A}\}, \{y_1, \dots, y_m \sqcup \dot{A}\}, \{y_1, \dots, y_m \sqcup \dot{A}\}) \end{aligned} \tag{4.12}$$

$$\begin{aligned} &\{x \sqcup A\} \subseteq \{y \sqcup B\} \to \\ &\qquad x = y \wedge A \subseteq \{y \sqcup B\} \\ &\qquad \vee\; x \neq y \wedge x \in B \wedge A \subseteq \{y \sqcup B\} \end{aligned} \tag{4.13}$$

$$\{x \sqcup A\} \subseteq [k, m] \to k \leq x \leq m \wedge A \subseteq [k, m] \tag{4.14}$$

$$[k, m] \subseteq A \to m < k \vee \dot{N} \subseteq A \wedge \mathit{size}(\dot{N}, m - k + 1) \tag{4.15}$$

$$[k, m] \subseteq [i, j] \to m < k \vee i \leq k \wedge m \leq j \tag{4.16}$$

Figure 13: Rewrite rules for $\subseteq$-constraints

Irreducible forms:

- $\dot{A} \subseteq \dot{B}$, $\dot{A}$ and $\dot{B}$ distinct variables
- $\dot{A} \subseteq \{y \sqcup B\}$.
- $\dot{A} \subseteq [k, m]$ and either k or m are variables

4.6 Intersection

Syntax: $inters(t_1, t_2, t_3)$.
Informal semantics: if t_1, t_2 and t_3 are sets, then $t_3 = t_1 \cap t_2$.
Rewrite rules: see Fig. 14-15.

If $x : \mathsf{U}$; $A, B, C : \mathsf{Set}$; $k, m, i, j : \mathsf{Int}$ then:

$$inters(\dot{A}, \dot{A}, B) \rightarrow A = B \tag{4.17}$$

$$inters(\emptyset, B, C) \rightarrow C = \emptyset \tag{4.18}$$

$$inters(A, \emptyset, C) \rightarrow C = \emptyset \tag{4.19}$$

$$inters(A, B, \emptyset) \rightarrow A \parallel B \tag{4.20}$$

If $B \notin \mathcal{V}$:

$$inters(\{x \sqcup A\}, B, \dot{C}) \rightarrow \tag{4.21}$$
$$x \in B \wedge \dot{C} = \{x \sqcup N_2\} \wedge inters(A, B, N_2) \vee x \notin B \wedge inters(A, B, \dot{C})$$

If $B \notin \mathcal{V}$:

$$\begin{aligned} & inters(B, \{x \sqcup A\}, \dot{C}) \rightarrow \\ & \qquad x \in B \wedge \dot{C} = \{x \sqcup N_2\} \wedge inters(A, B, N_2) \\ & \qquad \vee x \notin B \wedge inters(A, B, \dot{C}) \end{aligned} \tag{4.22}$$

If $A \not\equiv [\cdot, \cdot]$ and $B \not\equiv [\cdot, \cdot]$:

$$inters(A, B, \{x \sqcup C\}) \rightarrow \tag{4.23}$$
$$A = \{x \sqcup N_1\} \wedge B = \{x \sqcup N_2\} \wedge inters(N_1, N_2, C)$$

(4.24)

Figure 14: Rewrite rules for *inters*-constraints

If A, B, C : Set; k, m, i, j : Int then:

$$\begin{aligned}
&inters([k,m],[i,j],C) \rightarrow \\
&\quad j < i \wedge C = \emptyset \\
&\quad \vee\, j < k \wedge C = \emptyset \\
&\quad \vee\, m < i \wedge C = \emptyset \\
&\quad \vee\, m < k \wedge C = \emptyset \\
&\quad \vee\, k \leq i \wedge i \leq m \wedge m \leq j \wedge C = [i,m] \\
&\quad \vee\, i \leq k \wedge k \leq j \wedge j \leq m \wedge C = [k,j] \\
&\quad \vee\, k \leq m \wedge i \leq j \wedge k < i \wedge j < m \wedge C = [i,j] \\
&\quad \vee\, k \leq m \wedge i \leq j \wedge i < k \wedge m < j \wedge C = [k,m]
\end{aligned} \tag{4.25}$$

If $C \notin \mathcal{V}$ or ($A \notin \mathcal{V}$ and $B \notin \mathcal{V}$):

$$inters(A,B,C) \rightarrow un(C,\dot{N}_1,A) \wedge un(C,\dot{N}_2,B) \wedge \dot{N}_1 \parallel \dot{N}_2 \tag{4.26}$$

Figure 15: Rewrite rules for *inters*-constraints (second)

Irreducible forms:

- $inters(\dot{A}, B, \dot{C})$, $\dot{A}$ and B are not the same variable
- $inters(A, \dot{B}, \dot{C})$, A and $\dot{B}$ are not the same variable

4.7 Difference

Syntax: $diff(t_1, t_2, t_3)$.
Informal semantics: if t_1, t_2 and t_3 are sets, then $t_3 = t_1 \setminus t_2$.
Rewrite rules: see Fig. 16.

If $A, B, C : \mathsf{Set}$ then:

$$diff(A, B, C) \rightarrow \\ un(C, A, A) \wedge un(B, C, N) \wedge un(A, N, N) \wedge B \parallel C \tag{4.27}$$

Figure 16: Rewrite rules for *diff*-constraints

Irreducible form: none.

5 Rewrite rules for (positive) relational constraints

5.1 Identity

Syntax: $id(t_1, t_2)$.
Informal semantics: if t_1 is a set and t_2 is a binary relation, then t_2 is the identity relation over set t_1.
Rewrite rules: see Fig. 17.

If R, A : Set; x, y, x_i, y_i, a_i : U then:

$$id(\dot{R}, \dot{R}) \to \dot{R} = \emptyset \tag{id$_1$}$$

$$id(\emptyset, R) \to R = \emptyset \tag{id$_2$}$$

$$id(A, \emptyset) \to A = \emptyset \tag{id$_3$}$$

$$id(\{a_1, \dots, a_n \sqcup \dot{R}\}, \dot{R}) \to \mathit{false} \tag{id$_4$}$$

$$id(\dot{R}, \{(x_1, y_1), \dots, (x_n, y_n) \sqcup \dot{R}\}) \to \mathit{false} \tag{id$_5$}$$

$$\begin{aligned} &id(\{a_1, \dots, a_n \sqcup \dot{R}\}, \{(x_1, y_1), \dots, (x_m, y_m) \sqcup \dot{R}\}) \to \\ &\quad id(\{a_1, \dots, a_n\}, \{(x_1, y_1), \dots, (x_m, y_m)\}) \wedge \dot{R} = \emptyset \end{aligned} \tag{id$_6$}$$

$$id(\{x \sqcup A\}, R) \to R = \{(x, x) \sqcup N\} \wedge id(A, N) \tag{id$_7$}$$

$$id(A, \{(x, y) \sqcup R\}) \to x = y \wedge A = \{x \sqcup N\} \wedge id(N, R) \tag{id$_8$}$$

Figure 17: Rewrite rules for *id*-constraints

Irreducible form:

- $id(\dot{A}, \dot{R})$, $\dot{A}$ and $\dot{R}$ distinct variables.

5.2 Inverse

Syntax: $inv(t_1, t_2)$.
Informal semantics: if t_1 and t_2 are binary relations, then $t_2 = t_1^{\smile}$.
Rewrite rules: see Fig. 18.

If $R, S : \mathsf{Set}$; $x, y, x_i, y_i, a_i, b_i : \mathsf{U}$ then:

$$inv(R, \emptyset) \to R = \emptyset \tag{$1^{\smile}$}$$

$$inv(\emptyset, S) \to S = \emptyset \tag{$2^{\smile}$}$$

$$\begin{aligned}&inv(\dot{R}, \{(x_1, y_1), \dots, (x_n, y_n) \sqcup \dot{R}\}) \to \\ &\quad \dot{R} = \{(x_1, y_1), (y_1, x_1) \sqcup N\} \land inv(N, \{(x_2, y_2), \dots, (x_n, y_n) \sqcup N\})\end{aligned} \tag{$3^{\smile}$}$$

$$\begin{aligned}&inv(\{(x_1, y_1), \dots, (x_n, y_n) \sqcup \dot{S}\}, \dot{S}) \to \\ &\quad \dot{S} = \{(x_1, y_1), (y_1, x_1) \sqcup N\} \land inv(\{(x_2, y_2), \dots, (x_n, y_n) \sqcup N\}, N)\end{aligned} \tag{$4^{\smile}$}$$

$$\begin{aligned}&inv(\{(x_1, y_1), \dots, (x_n, y_n) \sqcup \dot{R}\}, \{(a_1, b_1), \dots, (a_m, b_m) \sqcup \dot{R}\}) \to \\ &\quad \{(y_1, x_1) \sqcup N_1\} = \{(a_1, b_1), \dots, (a_m, b_m)\} \\ &\qquad \land un(\dot{R}, N_1, N_2) \land inv(\{(x_2, y_2), \dots, (x_n, y_n) \sqcup \dot{R}\}, N_2) \\ &\quad \lor ((y_1, x_1) \notin \{(a_1, b_1), \dots, (a_m, b_m)\} \land (x_1, y_1) \notin \{(a_1, b_1), \dots, (a_m, b_m)\} \\ &\qquad \land \dot{R} = \{(x_1, y_1), (y_1, x_1) \sqcup N\} \land ((y_1, x_1) \notin \{(x_1, y_1), \dots, (x_n, y_n)\} \\ &\qquad\qquad \land inv(\{(x_2, y_2), \dots, (x_n, y_n) \sqcup N\}, \{(a_1, b_1), \dots, (a_m, b_m) \sqcup N\}) \\ &\qquad\quad \lor \{(y_1, x_1) \sqcup N_3\} = \{(x_2, y_2), \dots, (x_n, y_n)\} \land un(N, N_3, N_4) \\ &\qquad\qquad \land inv(N_4, \{(a_1, b_1), \dots, (a_m, b_m) \sqcup N\})) \\ &\quad \lor (y_1, x_1) \notin \{(a_1, b_1), \dots, (a_m, b_m)\} \\ &\qquad \land \{(x_1, y_1) \sqcup N_5\} = \{(a_1, b_1), \dots, (a_m, b_m)\} \\ &\qquad \land \dot{R} = \{(y_1, x_1) \sqcup N\} \land un(N, N_5, N_6) \\ &\qquad \land inv(\{(x_2, y_2), \dots, (x_n, y_n) \sqcup N\}, N_6)\end{aligned} \tag{$5^{\smile}$}$$

$$inv(R, \{(y, x) \sqcup S\}) \to R = \{(x, y) \sqcup N\} \land inv(N, S) \tag{$6^{\smile}$}$$

$$inv(\{(x, y) \sqcup R\}, S) \to S = \{(y, x) \sqcup N\} \land inv(R, N) \tag{$7^{\smile}$}$$

Figure 18: Rewrite rules for *inv*-constraints

Irreducible form:

- $inv(\dot{R}, \dot{S})$.

5.3 Composition

Syntax: $comp(t_1, t_2, t_3)$.
Informal semantics: if t_1, t_2 and t_3 are binary relations, then $t_3 = t_1 \circ t_2$.
Rewrite rules: see Fig. 19.

If $Q, R, S, T : \mathsf{Set}$; $Q \not\equiv \emptyset$; $t, u, x, z : \mathsf{U}$ then:

$$comp(\emptyset, S, T) \to T = \emptyset \qquad (\circ_1)$$

$$comp(R, \emptyset, T) \to T = \emptyset \qquad (\circ_2)$$

$$comp(\{(x,u)\}, \{(t,z)\}, T) \to (u = t \wedge T = \{(x,z)\}) \vee (u \neq t \wedge T = \emptyset) \qquad (\circ_3)$$

$$\begin{aligned} &comp(\{(x,u) \sqcup R\}, \{(t,z) \sqcup S\}, \emptyset) \to \\ &\quad u \neq t \\ &\quad \wedge\ comp(\{(x,u)\}, S, \emptyset) \wedge comp(R, \{(t,z)\}, \emptyset) \wedge comp(R, S, \emptyset) \end{aligned} \qquad (\circ_4)$$

$$\begin{aligned} &comp(\{(x,t) \sqcup R\}, \{(u,z) \sqcup S\}, \dot{T}) \to \\ &\quad comp(\{(x,t)\}, \{(u,z)\}, N_1) \\ &\quad \wedge\ comp(\{(x,t)\}, S, N_2) \wedge comp(R, \{(u,z)\}, N_3) \\ &\quad \wedge\ comp(R, S, N_4) \\ &\quad \wedge\ un(N_1, N_2, N_3, N_4, \dot{T}) \end{aligned} \qquad (\circ_5)$$

$$\begin{aligned} &comp(R, S, \{(x,z) \sqcup T\}) \to \\ &\quad un(N_x, N_{rt}, R) \wedge un(N_z, N_{st}, S) \\ &\quad N_x = \{(x,n) \sqcup N_1\} \wedge N_z = \{(n,z) \sqcup N_2\} \\ &\quad \wedge\ comp(\{(x,x)\}, N_1, N_1) \wedge comp(N_2, \{(z,z)\}, N_2) \\ &\quad \wedge\ comp(N_x, N_{st}, N_3) \wedge comp(N_{rt}, N_z, N_4) \wedge comp(N_{rt}, N_{st}, N_5) \\ &\quad \wedge\ un(N_3, N_4, N_5, T) \end{aligned} \qquad (\circ_6)$$

Figure 19: Rewrite rules for *comp*-constraints

Irreducible forms:

- $comp(\dot{R}, S, \dot{T})$, $S \not\equiv \emptyset$
- $comp(R, \dot{S}, \dot{T})$, $R \not\equiv \emptyset$
- $comp(\dot{R}, S, \emptyset)$
- $comp(R, \dot{S}, \emptyset)$

5.4 Domain

Syntax: $dom(t_1, t_2)$.
Informal semantics: if t_1 is a binary relation and t_2 is a set, then $t_2 = \mathrm{dom}\, t_1$.
Rewrite rules: see Fig. 20.

If $R, A : \mathsf{Set}$; $x, y : \mathsf{U}$ then:

$$dom(\dot{R}, \dot{R}) \to \dot{R} = \emptyset \tag{dom$_1$}$$

$$dom(R, \emptyset) \to R = \emptyset \tag{dom$_2$}$$

$$dom(\emptyset, A) \to A = \emptyset \tag{dom$_3$}$$

$$dom(\dot{R}, \{x \sqcup A\}) \to un(N_1, N_2, \dot{R}) \wedge dom(N_1, \{x\}) \wedge dom(N_2, A) \tag{dom$_4$}$$

$$dom(\dot{R}, \{x\})\} \to comp(\{(x, x)\}, \dot{R}, \dot{R}) \wedge \dot{R} \neq \emptyset \tag{dom$_5$}$$

$$dom(\{(x, y) \sqcup R\}, A) \to A = \{x \sqcup N_1\} \wedge dom(R, N_1) \tag{dom$_6$}$$

Figure 20: Rewrite rules for *dom*-constraints

Irreducible form:

- $dom(\dot{R}, \dot{A})$, $\dot{R}$ and $\dot{A}$ distinct variables.

5.5 Range

Syntax: $ran(t_1, t_2)$.
Informal semantics: if t_1 is a binary relation and t_2 is a set, then $t_2 = \operatorname{ran} t_1$.
Rewrite rules: see Fig. 21.

If R, A : Set; x, y : U then:

$$ran(\dot{R}, \dot{R}) \to \dot{R} = \emptyset \qquad (\text{ran}_7)$$

$$ran(R, \emptyset) \to R = \emptyset \qquad (\text{ran}_8)$$

$$ran(\emptyset, A) \to A = \emptyset \qquad (\text{ran}_9)$$

$$ran(\dot{R}, \{y \sqcup A\}) \to un(N_1, N_2, \dot{R}) \wedge ran(N_1, \{y\}) \wedge ran(N_2, A) \qquad (\text{ran}_{10})$$

$$ran(\dot{R}, \{y\})\} \to comp(\dot{R}, \{(y, y)\}, \dot{R}) \wedge \dot{R} \neq \emptyset \qquad (\text{ran}_{11})$$

$$ran(\{(x, y) \sqcup R\}, A) \to A = \{y \sqcup N_1\} \wedge ran(R, N_1) \qquad (\text{ran}_{12})$$

Figure 21: Rewrite rules for *ran*-constraints

Irreducible form:

- $ran(\dot{R}, \dot{A})$, $\dot{R}$ and $\dot{A}$ distinct variables.

5.6 Other relational constraints

The following are relational constraints that can be expressed as $\mathcal{BR}$-formulas (i.e. by quantifier-free first order formulas). For each of them there is just a single rewrite rule replacing the constraint with the corresponding $\mathcal{BR}$-formula.

If R, S, T, A, B, C, f : Set; x, y : U then:

$$ran(R, A) \rightarrow inv(R, N) \wedge dom(N, A) \tag{5.1}$$

$$\begin{aligned} &dres(A, R, S) \rightarrow \\ &\quad un(S, N_1, R) \wedge dom(S, N_2) \wedge N_2 \subseteq A \wedge dom(N_1, N_3) \wedge A \parallel N_3 \end{aligned} \tag{5.2}$$

$$\begin{aligned} &rres(R, A, S) \rightarrow \\ &\quad un(S, N_1, R) \wedge ran(S, N_2) \wedge N_2 \subseteq A \wedge ran(N_1, N_3) \wedge A \parallel N_3 \end{aligned} \tag{5.3}$$

$$\begin{aligned} &dares(A, R, S) \rightarrow \\ &\quad dres(A, R, T) \wedge un(S, T, R) \wedge S \parallel T \end{aligned} \tag{5.4}$$

$$\begin{aligned} &rares(R, A, S) \rightarrow \\ &\quad rres(R, A, T) \wedge un(S, T, R) \wedge S \parallel T \end{aligned} \tag{5.5}$$

$$rimg(R, A, B) \rightarrow dres(A, R, N) \wedge ran(N, B) \tag{5.6}$$

$$oplus(R, S, T) \rightarrow dom(S, N_1) \wedge dares(N_1, R, N_2) \wedge un(N_2, S, T) \tag{5.7}$$

$$apply(f, x, y) \rightarrow (x, y) \in f \wedge pfun(f) \tag{5.8}$$

$$\begin{aligned} &cp(A, B, R) \mathrel{\widehat{=}} \\ &\quad dom(N_1, A) \wedge ran(N_1, N_2) \wedge N_2 \subseteq \{n\} \\ &\quad \wedge dom(N_2, B) \wedge ran(N_2, N_3) \wedge N_3 \subseteq \{n\} \\ &\quad \wedge inv(N_2, N_4) \wedge comp(N_1, N_4, R) \end{aligned} \tag{5.9}$$

Figure 22: Rewrite rules for other relational constraints

5.7 Specialized rewrite rules for partial functions

The rewrite rules specialized for partial functions are listed in Figures 28 to 24.

Domain of partial functions

Syntax: $dom(t_1, t_2)$.
Informal semantics: if t_1 is a partial function and t_2 is a set, then $t_2 = \operatorname{dom} t_1$.
Rewrite rules: see Fig. 23.

If $f, A : \mathsf{Set};\ x, y : \mathsf{U}$ then:

$$dom(\dot{f}, \dot{f}) \to \dot{f} = \emptyset \qquad (\text{dom}_{13})$$

$$dom(f, \emptyset) \to f = \emptyset \qquad (\text{dom}_{14})$$

$$dom(\emptyset, A) \to A = \emptyset \qquad (\text{dom}_{15})$$

$$dom(\dot{f}, \{x \sqcup A\}) \to \dot{f} = \{(x, n) \sqcup N\} \wedge dom(N, A) \qquad (\text{dom}_{16})$$

$$dom(\{(x, y) \sqcup f\}, A) \to A = \{x \sqcup N\} \wedge dom(f, N) \qquad (\text{dom}_{17})$$

Figure 23: Rewrite rules for *dom*-constraints over partial functions

Irreducible form:

- $dom(\dot{f}, \dot{A})$, $\dot{f}$ and $\dot{A}$ distinct variables.

Composition of partial functions

Syntax: $comp(t_1, t_2, t_3)$.
Informal semantics: if t_1, t_2 and t_3 are partial functions, then $t_3 = t_1 \circ t_2$.
Rewrite rules: see Fig. 24.

If $f, g, h, A : \mathsf{Set}$; $x, y, z : \mathsf{U}$ then:

$$comp(\emptyset, g, h) \rightarrow h = \emptyset \tag{$\circ_7$}$$

$$comp(f, \emptyset, h) \rightarrow h = \emptyset \tag{$\circ_8$}$$

$$comp(f, g, \emptyset) \rightarrow ran(f, N_1) \wedge dom(g, N_2) \wedge N_1 \parallel N_2 \tag{$\circ_9$}$$

$$\begin{aligned} & comp(\{(x, y) \sqcup f\}, g, \dot{h})\} \rightarrow \\ & \qquad g = \{(y, n) \sqcup N_1\} \wedge \dot{h} = \{(x, n) \sqcup N_2\} \wedge comp(f, g, N_2) \\ & \qquad \vee\ dom(g, N_1) \wedge y \notin N_1 \wedge comp(f, g, \dot{h}) \end{aligned} \tag{$\circ_{10}$}$$

$$\begin{aligned} & comp(f, g, \{(x, z) \sqcup h\}) \rightarrow \\ & \qquad f = \{(x, n) \sqcup N_1\} \wedge g = \{(n, z) \sqcup N_2\} \wedge comp(N_1, g, h) \end{aligned} \tag{$\circ_{11}$}$$

Figure 24: Rewrite rules for *comp*-constraints over partial functions

Irreducible forms:

- $comp(\dot{f}, g, \dot{h})$, $g \not\equiv \emptyset$.
- $comp(f, \dot{g}, \dot{h})$, $f \not\equiv \emptyset$.

6 Rewrite rules for sort constraints

Sort constraint *pair*

Syntax: $pair(t)$.
Informal semantics: t is a pair.
Rewrite rules: see Fig. 25.

If $t : \mathsf{U}$ then:

$$pair(t) \to t = (n_1, n_2) \qquad (\text{pair}_1)$$

Figure 25: Rewrite rules for *pair*-constraints

Irreducible form: none.

Sort constraint *set*

Syntax: $set(t)$.
Informal semantics: t is a set.
Rewrite rules: see Fig. 26.

If $A : \mathsf{Set}$; $t_1, k, m : \mathsf{U}$; $t_2 : \mathsf{O}$ then:

$$set(\emptyset) \to true \qquad (\text{set}_1)$$

$$set(\{t_1 \sqcup A\}) \to set(A) \qquad (\text{set}_2)$$

$$set(t_2) \to false \qquad (\text{set}_3)$$

$$set([k, m]) \to integer(k) \wedge integer(m) \qquad (\text{set}_4)$$

Figure 26: Rewrite rules for *set*-constraints

Irreducible form:

- $set(\dot{x})$.

Sort constraint *rel*

Syntax: $rel(t)$.
Informal semantics: if t is a set, then t is a binary relation.
Rewrite rules: see Fig. 27.

If R : Set; t, k, m : U then:

$$rel(\emptyset) \rightarrow true \quad (\leftrightarrow_1)$$

$$rel(\{t \sqcup R\}) \rightarrow t = (n_1, n_2) \wedge rel(R) \quad (\leftrightarrow_2)$$

$$rel([k, m]) \rightarrow m < k \quad (\leftrightarrow_3)$$

Figure 27: Rewrite rules for *rel*-constraints

Irreducible form:

- $rel(\dot{x})$.

Sort constraint *pfun*

Syntax: $pfun(t)$.
Informal semantics: if t is a set, then t is a partial function.
Rewrite rules: see Fig. 28.

If f : Set; t, k, m : U then:

$$pfun(\emptyset) \rightarrow true \quad (\nrightarrow_4)$$

$$pfun(\{t \sqcup f\}) \rightarrow t = (n_1, n_2) \wedge comp(\{(n_1, n_1)\}, f, \emptyset) \wedge pfun(f) \quad (\nrightarrow_5)$$

$$pfun([k, m]) \rightarrow m < k \quad (\nrightarrow_6)$$

Figure 28: Rewrite rules for *pfun*-constraints

Irreducible form:

- $pfun(\dot{x})$.

Sort constraint *npair*

Syntax: $npair(t)$.
Informal semantics: t is not a pair.
Rewrite rules: see Figure 29.

If $t_1, \ldots, t_n, f(t_1, \ldots, t_n) : \mathsf{U}$, $n \geq 0$ then:

$$npair((t_1, t_2)) \rightarrow false \qquad (\text{pair}_2)$$

$$\text{If } f \not\equiv (\cdot,\cdot){:}npair(f(t_1, \ldots, t_n)) \rightarrow true \qquad (\text{pair}_3)$$

$$\text{If } n \not\equiv 2{:}npair(f(t_1, \ldots, t_n)) \rightarrow true \qquad (\text{pair}_4)$$

Figure 29: Rewrite rules for negative *pair*-constraints

Irreducible forms:

- $npair(\dot{x})$.

Sort constraint *nset*

Syntax: $nset(t)$.
Informal semantics: t is not a set.
Rewrite rules: see Figures 30 and 29.

If $A : \mathsf{Set}$; $t_1, k, m : \mathsf{U}$; $t_2 : \mathsf{O}$ then:

$$nset(\emptyset) \rightarrow false \qquad (\text{set}_5)$$

$$nset(\{t_1 \sqcup A\}) \rightarrow false \qquad (\text{set}_6)$$

$$nset(t_2) \rightarrow true \qquad (\text{set}_7)$$

$$nset([k, m]) \rightarrow ninteger(k) \vee ninteger(m) \qquad (\text{set}_8)$$

Figure 30: Rewrite rules for negative *set*-constraints

Irreducible forms:

- $nset(\dot{x})$.

Sort constraints *nrel* **and** *npfun*

Syntax: $nrel(t)$, $npfun(t)$.
Informal semantics: t is not a relation, t is not a partial function.
Rewrite rules: see Fig. 37.

If R, S, T, A, f : Set then:

$$nrel(R) \to n \in R \wedge npair(n) \qquad (\leftrightarrow_4)$$

$$npfun(f) \to (n_1, n_2) \in f \wedge (n_1, n_3) \in f \wedge n_2 \neq n_3 \vee nrel(f) \qquad (\nrightarrow_7)$$

Figure 31: Rewrite rules for negative relational base constraints

Irreducible forms: none.

7 Rewrite rules for negative set constraints

7.1 Not membership

Syntax: $t_1 \notin t_2$.
Informal semantics: if t_2 is a set, then t_1 is not a member of t_2.
Rewrite rules: see Fig. 32.

If $x, y : \mathsf{U}$; $A : \mathsf{Set}$; $k, m : \mathsf{Int}$ then:

$$x \notin \emptyset \to \mathit{true} \qquad (\in_5)$$

$$x \notin \{y \sqcup A\} \to x \neq y \wedge x \notin A \qquad (\in_6)$$

$$\text{If } \dot{A} \in \mathit{vars}(x): x \notin \dot{A} \to \mathit{true} \qquad (\in_7)$$

$$x \notin [k, m] \to \mathit{ninteger}(x) \vee x < k \vee m < x \qquad (\in_8)$$

Figure 32: Rewrite rules for $\notin$-constraints

Irreducible form:

- $t \notin \dot{A}$ and $\dot{A}$ does not occur in t.

7.2 Not union

Syntax: $nun(t_1, t_2, t_3)$.
Informal semantics: if t_1, t_2 and t_3 are sets, then $t_3 \neq t_1 \cup t_2$.
Rewrite rules: see Fig. 33.

If A, B, C : Set then:

$$nun(A, B, C) \rightarrow \\ N \in C \wedge N \notin A \wedge N \notin B \\ \vee N \in A \wedge N \notin C \\ \vee N \in B \wedge N \notin C \qquad (\cup_{15})$$

Figure 33: Rewrite rules for negative *un*-constraints

Irreducible form: none.

Not disjoint

Syntax: $t_1 \not\parallel t_2$.
Informal semantics: if t_1 and t_2 are sets, then $t_1 \cap t_2 \neq \emptyset$.
Rewrite rules: see Fig. 34.

If A, B : Set then:

$$A \not\parallel B \rightarrow n \in A \wedge n \in B \qquad (\parallel_{10})$$

Figure 34: Rewrite rules for negative $\parallel$-constraints

Irreducible form: none.

Not size (not set cardinality)

Syntax: $nsize(t_1, t_2)$.
Informal semantics: if t_1 is a set and t_2 is an integer term, then t_2 is not the cardinality of t_1.
Rewrite rules: see Fig. 35.

If A : Set; k, m, p : Int then:

$$nsize([k, m], p) \rightarrow (m < k \wedge p \neq 0) \vee (k \leq m \wedge p \neq m - k + 1) \tag{7.11}$$

$$nsize(A, p) \rightarrow size(A, n) \wedge n \neq p \tag{7.12}$$

Figure 35: Rewrite rules for negative *size*-constraints

Irreducible form: none.

7.3 Other negative set constraints

If R, S, T, A, B, C, f : Set then:

$$nsubset(A, B) \rightarrow n \in A \wedge n \notin B \tag{7.1}$$

$$\begin{aligned} & ninters(A, B, C) \rightarrow \\ & \qquad n \in C \wedge (n \notin A \vee n \notin B) \\ & \qquad \vee\ n \in A \wedge n \in B \wedge n \notin C \end{aligned} \tag{7.2}$$

$$\begin{aligned} & ndiff(A, B, C) \rightarrow \\ & \qquad n \in C \wedge n \notin A \vee n \in C \wedge n \in B \\ & \qquad \vee\ n \notin C \wedge n \in A \wedge n \notin B \end{aligned} \tag{7.3}$$

Figure 36: Rewrite rules for other negative set constraints

Irreducible form: none.

8 Rewrite rules for negative relational constraints

If $R, S, T, A : \mathsf{Set}$ then:

$$
\begin{aligned}
& \mathit{ndom}(R, A) \rightarrow \\
& \qquad (n_1, n_2) \in R \wedge n_1 \notin A \\
& \qquad \vee\ n_1 \in A \wedge \mathit{comp}(\{(n_1, n_1)\}, R, \emptyset) \\
& \qquad \vee\ \mathit{nrel}(R)
\end{aligned}
\tag{dom$_{18}$}
$$

$$
\begin{aligned}
& \mathit{ninv}(R, S) \rightarrow \\
& \qquad (n_1, n_2) \in R \wedge (n_2, n_1) \notin S \\
& \qquad \vee\ (n_1, n_2) \notin R \wedge (n_2, n_1) \in S \\
& \qquad \vee\ \mathit{nrel}(R) \vee \mathit{nrel}(S)
\end{aligned}
\tag{$8^{\smile}$}
$$

$$
\begin{aligned}
& \mathit{ncomp}(R, S, T) \rightarrow \\
& \qquad (n_1, n_2) \in R \wedge (n_2, n_3) \in S \wedge (n_1, n_3) \notin T \\
& \qquad \vee (n_1, n_3) \in T \\
& \qquad\quad \wedge\ \mathit{comp}(\{(n_1, n_1)\}, R, N_1) \\
& \qquad\quad \wedge\ \mathit{comp}(S, \{(n_3, n_3)\}, N_2) \wedge \mathit{comp}(N_1, N_2, \emptyset) \\
& \qquad \vee\ \mathit{nrel}(R) \vee \mathit{nrel}(S) \vee \mathit{nrel}(T)
\end{aligned}
\tag{$\circ_{12}$}
$$

$$
\begin{aligned}
& \mathit{nid}(A, f) \rightarrow \\
& \qquad n_1 \in A \wedge (n_1, n_1) \notin f \\
& \qquad \vee\ n_1 \notin A \wedge (n_1, n_1) \in f \\
& \qquad \vee\ n_1 \neq n_2 \wedge (n_1, n_2) \in f \\
& \qquad \vee\ n \in f \wedge \mathit{npair}(n)
\end{aligned}
\tag{id$_{13}$}
$$

Figure 37: Rewrite rules for negative relational constraints

Irreducible form: none.

If R, S, T, A, B, C, f : Set then:

$$
\begin{aligned}
& nran(R, A) \rightarrow \\
& \qquad (n_1, n_2) \in R \wedge n_2 \notin A \\
& \qquad \vee\ n_1 \in A \wedge comp(R, \{(n_1, n_1)\}, \emptyset) \\
& \qquad \vee\ nrel(R)
\end{aligned} \tag{8.1}
$$

$$
\begin{aligned}
& ndres(A, R, S) \rightarrow \\
& \qquad (n_1, n_2) \in S \wedge n_1 \notin A \\
& \qquad \vee\ (n_1, n_2) \in S \wedge (n_1, n_2) \notin R \\
& \qquad \vee\ (n_1, n_2) \in R \wedge n_1 \in A \wedge (n_1, n_2) \notin S \\
& \qquad \vee\ nrel(R) \vee nrel(S)
\end{aligned} \tag{8.2}
$$

$$
\begin{aligned}
& nrres(R, A, S) \rightarrow \\
& \qquad (n_1, n_2) \in S \wedge n_2 \notin A \\
& \qquad \vee\ (n_1, n_2) \in S \wedge (n_1, n_2) \notin R \\
& \qquad \vee\ (n_1, n_2) \in R \wedge n_2 \in A \wedge (n_1, n_2) \notin S \\
& \qquad \vee\ nrel(R) \vee nrel(S)
\end{aligned} \tag{8.3}
$$

$$
\begin{aligned}
& ndares(A, R, S) \rightarrow \\
& \qquad (n_1, n_2) \in S \wedge n_1 \in A \\
& \qquad \vee\ (n_1, n_2) \in S \wedge (n_1, n_2) \notin R \\
& \qquad \vee\ (n_1, n_2) \in R \wedge n_1 \notin A \wedge (n_1, n_2) \notin S \\
& \qquad \vee\ nrel(R) \vee nrel(S)
\end{aligned} \tag{8.4}
$$

$$
\begin{aligned}
& nrares(R, A, S) \rightarrow \\
& \qquad (n_1, n_2) \in S \wedge n_2 \in A \\
& \qquad \vee\ (n_1, n_2) \in S \wedge (n_1, n_2) \notin R \\
& \qquad \vee\ (n_1, n_2) \in R \wedge n_2 \notin A \wedge (n_1, n_2) \notin S \\
& \qquad \vee\ nrel(R) \vee nrel(S)
\end{aligned} \tag{8.5}
$$

$$
napply(f, x, y) \rightarrow (x, y) \notin f \vee npfun(f) \tag{8.6}
$$

$$
\tag{8.7}
$$

Figure 38: Rewrite rules for negative relational constraints—cont'd

If R, S, T, A, B, C, f : Set then:

$$
\begin{aligned}
& nrimg(R, A, B) \rightarrow \\
& \qquad n_1 \in B \wedge id(A, N_1) \wedge comp(N_1, R, N_2) \wedge comp(N_2, \{(n_1, n_1)\}, \emptyset) \\
& \qquad \vee\, n_1 \notin B \wedge (n_2, n_1) \in R \wedge n_2 \in A \\
& \qquad \vee\, nrel(R)
\end{aligned} \tag{8.8}
$$

$$
\begin{aligned}
& noplus(R, S, T) \rightarrow \\
& \qquad (n_1, n_2) \in T \wedge (n_1, n_2) \notin S \wedge (n_1, n_2) \notin R \\
& \qquad \vee\, (n_1, n_2) \in T \\
& \qquad\quad \wedge\, (n_1, n_2) \notin S \\
& \qquad\quad \wedge\, (n_1, n_2) \in R \wedge comp(\{(n_1, n_1)\}, S, \{(n_1, n_3) \sqcup N\}) \\
& \qquad \vee\, (n_1, n_2) \notin T \wedge (n_1, n_2) \in S \\
& \qquad \vee\, (n_1, n_2) \notin T \wedge (n_1, n_2) \in R \wedge comp(\{(n_1, n_1)\}, S, \emptyset) \\
& \qquad \vee\, nrel(R) \vee nrel(S) \vee nrel(T)
\end{aligned} \tag{8.9}
$$

$$
\begin{aligned}
& ncp(A, B, R) \rightarrow \\
& \quad n_1 \in A \wedge n_2 \in B \wedge (n_1, n_2) \notin R \\
& \quad \vee\, (n_1 \notin A \vee n_2 \notin B) \wedge (n_1, n_2) \in R
\end{aligned} \tag{8.10}
$$

Figure 39: Rewrite rules for negative relational constraints—cont'd

9 The solver

The global organization of the solver for $\mathcal{L}_{\{\cdot\}}$, called $SAT_{\{\cdot\}}$, is shown in Algorithm 1.

Algorithm 1 The $SAT_{\{\cdot\}}$ solver. Φ is the input formula.

```
Φ ← sort_infer(Φ);
Φ ← gen_size_leq(Φ);
repeat
    Φ′ ← Φ;
    Φ ← remove_neq(step_loop(Φ))
until Φ = Φ′;                                        [end of main loop]
let Φ be Φ_S ∧ Φ_⊆[];
let Φ_S be Φ_1 ∧ Φ_2;
if Φ_⊆[] ≠ true then
    return step_loop(solve_size_minsol(Φ_1) ∧ Φ_2 ∧ Φ_⊆[])
else
    return solve_size(Φ_1) ∧ Φ_2
end if
procedure step_loop(Φ)
    repeat
        Φ′ ← Φ;
        Φ ← STEP(Φ)                                  [STEP is a key procedure]
    until Φ = Φ′
    return Φ
end procedure
```

After the main loop, Φ is divided into $\Phi_{\subseteq[\,]}$ and Φ_S: $\Phi_{\subseteq[\,]}$ is a conjunction of constraints of the form $X \subseteq [p, q]$ where p or q are variables; Φ_S is the rest of Φ. In turn, Φ_S is divided into Φ_1 and Φ_2: Φ_1 contains all the integer constraints and all the *un*, $\parallel$ and *size* constraints, and Φ_2 is the rest of Φ_S (i.e., $\notin$-constraints, and $=$ and $\neq$ constraints not involving integer terms[1]).

Procedure **sort_infer**

The procedure sort_infer applies all possible sort inference rules (see Sect. 2) to all constraints occurring in the input formula Φ.

Procedure **gen_size_leq**

gen_size_leq simply adds integer constraints to the input formula Φ to force the second argument of *size*-constraints in Φ to be non-negative integers.

[1] If in Φ there are *size* constraints and in Φ_2 there are constraints beyond $\notin, =, \neq$, e.g. relational constraints, then Φ it's outside the decision procedure and thus the answer returned by $SAT_{\{\cdot\}}$ is unreliable.

Procedure remove_neq

The procedure remove_neq deals with the elimination of $\neq$-constraints involving set variables. The rewrite rules applied by remove_neq are described by the generic rule scheme of Figure 40.

If $X \in \mathcal{V}_{Set}$; $t : \mathsf{U}$; Φ is the input formula then:

If X occurs as an argument of a constraint $\pi(\dots)$ in Φ,

$$\pi \in \{un, \subseteq, inters, id, inv, comp\}, X \neq t \rightarrow \qquad (\neq_{15})$$

$$(n \in X \wedge n \notin t) \vee (n \in t \wedge n \notin X) \vee (X = \emptyset \wedge t \neq \emptyset)$$

Figure 40: Rule scheme for $\neq$-constraint elimination

Remark 2 *The third disjunct in the rule scheme of Figure 40 is necessary when t is a non-set term (in particular, when t is a variable which is subsequently bound to a non-set term). In this case the second disjunct is false while the first disjunct forces X to contain an element n; so we would miss the solution $X = \emptyset$ which conversely is obtained through the third disjunct.*

Procedure STEP

The key part of the solver $SAT_{\{\cdot\}}$ is the procedure STEP. STEP is a function that takes as its input a $\{\cdot\}$-formula Φ and returns a new $\{\cdot\}$-formula Φ'. The overall structure of STEP is shown in Figure 41.

$$
\begin{aligned}
&\mathsf{STEP}(\Phi): \\
&\qquad \text{let } \Phi \text{ be } \Phi_1 \vee \cdots \vee \Phi_n,\ n \geq 1; \\
&\qquad \Phi \to \mathsf{STEP}_0(\Phi_1) \\
&\qquad\qquad \vee\ \mathsf{STEP}_0(\Phi_2) \\
&\qquad\qquad \dots \\
&\qquad\qquad \vee\ \mathsf{STEP}_0(\Phi_n);
\end{aligned}
\tag{9.16}
$$

$$
\begin{aligned}
&\mathsf{STEP}_0(\Phi): \\
&\qquad \text{for all } \pi \text{ in } \Pi\text{: } \Phi \leftarrow \mathsf{rw}_\pi(\Phi); \\
&\qquad \Phi \leftarrow \mathsf{sort\_check}(\mathsf{sort\_infer}(\Phi)); \\
&\qquad \mathsf{return}\ \Phi;
\end{aligned}
\tag{9.17}
$$

Figure 41: The procedure STEP

where $\Phi \to \mathsf{STEP}_0(\Phi_1) \vee \cdots \vee \mathsf{STEP}_0(\Phi_n)$ generalizes the definition of rewrite rule given in Section 1 by allowing the left-hand side to be any $\{\cdot\}$-formula; hence, Φ is non-deterministically rewritten to any of the formulas resulting from calling $\mathsf{STEP}_0(\Phi_i)$, $i \in 1..n$.

Procedure solve_size

solve_size is the adaptation of the decision procedure proposed by C. Zarba[2] for cardinality (*size*) constraints to our CLP framework. Both STEP and solve_size use the SWI-Prolog CLP(Q) library to solve linear integer arithmetic problems. These problems may be part of Φ or they are generated during set processing. CLP(Q) provides the library predicate `bb_inf`[3] to implement a decision procedure for linear integer arithmetic. Besides, solve_size uses a SAT solver implemented in Prolog by Howe and King[4] to solve a key aspect of Zarba's algorithm. $\mathsf{solve\_size}_{\mathsf{minsol}}(\Phi_1)$ is the call to solve_size in *minimal solution mode*, i.e., asking solve_size to compute the minimal solution of formula Φ_1—which in turn is implemented with a suitable call to `bb_inf`.

[2] C. G. Zarba, Combining sets with integers, in: A. Armando (Ed.), Fron- tiers of Combining Systems, 4th International Workshop, FroCoS 2002, Santa Margherita Ligure, Italy, April 8-10, 2002, Proceedings, Vol. 2309 of Lecture Notes in Computer Science, Springer, 2002, pp. 103–116. doi:10.1007/3-540-45988-X_9. URL `https://doi.org/10.1007/3-540-45988-X_9`

[3] `bb_inf`(*Vars*, *Expr*, *Min*, *Vert*) finds the vertex (*Vert*) of the minimum (*Min*) of the expression *Expr* subjected to the integer constraints present in the constraint store and assuming all the variables in *Vars* take integers values.

[4] J. M. Howe, A. King, A pearl on SAT and SMT solving in Prolog, Theor. Comput. Sci. 435 (2012) 43–55. doi:10.1016/j.tcs.2012.02.024. URL `https://doi.org/10.1016/j.tcs.2012.02.024`

Generic procedure rw_π

The procedure rw_π (see Figure 42) is a generic procedure, parametric with respect to the $\{\cdot\}$-constraint predicate symbol π. For each $\pi \in \Pi$, rw_π implements a *rewriting procedure for* π. rw_π takes as its input a $\{\cdot\}$-formula Φ and returns a new $\{\cdot\}$-formula Φ' which is obtained from Φ by repeatedly applying to it all possible rewrite rules for π. A rewrite rule $C \to \Phi$ is *applicable* to a constraint D if D matches C. Applying an applicable rule $C \to \Phi$ to a constraint D occurring in a formula Ψ causes D to be replaced by Φ in Ψ.

```
rw_π(Φ) :
    if Φ contains false then return false;
    else
        repeat
            select any π-constraint c in Φ;                                    (9.1)
            apply any applicable rewrite rule to c
        until there is no applicable rule for any π-constraint in Φ;
    return Φ;
```

Figure 42: Rewriting procedure for π-constraints

Note that if any of the rewrite rules called within STEP rewrites its input constraint to *false*, then the whole formula Φ is rewritten to *false*. In this case, a fixpoint is immediately detected, since STEP(*false*) returns *false*.

Procedure sort_check

The procedure sort_check applies the rewrite rules for sort checking to all possible pairs of sort constraints in Φ. If no pair exists for which the rules apply, then Φ is returned unchanged. Otherwise Φ is rewritten to *false*. The rewrite rules applied by sort_check are described by the generic rule scheme of Figure 43.

Let p be a predicate symbol in $\{set, rel, pfun\}$. If $x \in \mathcal{V}$ then:

$$nset(x) \wedge p(x) \to false \qquad (\mathrm{set}_9)$$

Figure 43: Rule scheme for sort consistency checking