Rewrite Rules for a Solver for Sets, Binary Relations and Partial Functions

Maximiliano Cristiá
Universidad Nacional de Rosario

Gianfranco Rossi
Università di Parma

January 9, 2019

Abstract

This document lists in a compact way all the rewrite rules used in the constraint solver for $\mathcal{L}_{\mathcal{BR}}$, a constraint language which provides both extensional finite sets and binary relations, along with basic operations on them. The constraint solver for $\mathcal{L}_{\mathcal{BR}}$ takes the form of a rewrite system acting on $\mathcal{BR}$-formulas, i.e., quantifier-free conjunctions and disjunctions of positive and negative $\mathcal{BR}$-constraints. A $\mathcal{BR}$-*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:

- $\{=\}$ (equality constraints)
- $\{\in, \mathit{un}, \|, \subseteq, \mathit{inters}, \mathit{diff}\}$ ((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)
- $\{\notin, \mathit{nun}, \nparallel, \mathit{nsubset}, \mathit{ninters}, \mathit{ndiff}\}$ (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)
- $\{\mathit{pair}, \mathit{set}, \mathit{rel}, \mathit{pfun}, \mathit{npair}, \mathit{nset}, \mathit{nrel}, \mathit{npfun}\}$ (sort constraints).

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

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

$$\phi \to \Phi$$

where ϕ is a π-constraint and Φ is a $\mathcal{BR}$-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*).

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

Notational conventions.

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

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:

$$t \in u \rightarrow t \in u \wedge set(u) \qquad (\inf_1)$$

$$un(t, u, v) \rightarrow set(t) \wedge set(u) \wedge set(v) \qquad (\inf_2)$$

$$t \parallel u \rightarrow set(t) \wedge set(u) \qquad (\inf_3)$$

$$t \subseteq u \rightarrow t \subseteq u \wedge set(t) \wedge set(u) \qquad (\inf_4)$$

$$inters(t, u, v) \rightarrow inters(t, u, v) \wedge set(t) \wedge set(u) \wedge set(v) \qquad (\inf_5)$$

$$diff(t, u, v) \rightarrow diff(t, u, v) \wedge set(t) \wedge set(u) \wedge set(v) \qquad (\inf_6)$$

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:

$$inv(t,u) \rightarrow inv(t,u) \wedge rel(t) \wedge rel(u) \quad (\inf_7)$$

$$comp(t,u,v) \rightarrow comp(t,u,v) \wedge rel(t) \wedge rel(u) \wedge rel(v) \quad (\inf_8)$$

$$id(t,u) \rightarrow id(t,u) \wedge set(t) \wedge rel(u) \quad (\inf_9)$$

$$dom(t,u) \rightarrow dom(t,u) \wedge rel(t) \wedge set(u) \quad (\inf_{10})$$

$$ran(t,u) \rightarrow ran(t,u) \wedge rel(t) \wedge set(u) \quad (\inf_{11})$$

$$dres(t,u,v) \rightarrow dres(t,u,v) \wedge set(t) \wedge rel(u) \wedge rel(v) \quad (\inf_{12})$$

$$rres(t,u,v) \rightarrow rres(t,u,v) \wedge rel(t) \wedge set(u) \wedge rel(v) \quad (\inf_{13})$$

$$dares(t,u,v) \rightarrow dares(t,u,v) \wedge set(t) \wedge rel(u) \wedge rel(v) \quad (\inf_{14})$$

$$rares(t,u,v) \rightarrow rares(t,u,v) \wedge rel(t) \wedge set(u) \wedge rel(v) \quad (\inf_{15})$$

$$rimg(t,u,v) \rightarrow rimg(t,u,v) \wedge rel(t) \wedge set(u) \wedge set(v) \quad (\inf_{16})$$

$$oplus(t,u,v) \rightarrow oplus(t,u,v) \wedge rel(t) \wedge rel(u) \wedge rel(v) \quad (\inf_{17})$$

$$apply(t,u,v) \rightarrow apply(t,u,v) \wedge rel(t) \quad (\inf_{18})$$

$$cp(t,u,v) \rightarrow cp(t,u,v) \wedge set(t) \wedge set(u) \wedge rel(v) \quad (\inf_{19})$$

$$rel(t) \rightarrow rel(t) \wedge set(t) \quad (\inf_{20})$$

$$pfun(t) \rightarrow pfun(t) \wedge rel(t) \quad (\inf_{21})$$

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 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.

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

Figure 3: Finding set terms

Remark 1 *$\mathcal{L}_{\mathcal{BR}}$ 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. 6), while literal $set(1)$ is rewritten to false by the rewrite rules for set-constraints (see Fig. 21); 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 ($vars(t_1, \ldots, t_n)$ denotes the set of variables occurring in $t_1, \ldots, t_n$).

If $x, y, t, t_i, u_i : \mathsf{U}$; $A, B : \mathsf{Set}$ then:

$$\dot{x} = \dot{x} \rightarrow \mathit{true} \tag{$=_1$}$$

$$\text{If } t \notin \mathcal{V}, t = \dot{x} \rightarrow \dot{x} = t \tag{$=_2$}$$

$$\text{If } \dot{A} \notin vars(t_1, \ldots, t_n),\ \dot{A} = \{t_1, \ldots, t_n \sqcup \dot{A}\} \rightarrow \dot{A} = \{t_1, \ldots, t_n \sqcup N\} \tag{$=_3$}$$

$$\text{If } \dot{x} \in vars(t),\ \dot{x} = t \rightarrow \mathit{false} \tag{$=_4$}$$

$$\begin{array}{l}\text{If } \dot{x} \text{ occurs in other literals of the input formula}, \dot{x} = t \rightarrow \\ \quad \dot{x} = t, \text{and substitute } \dot{x} \text{ by } t \text{ in all other literals}\end{array} \tag{$=_5$}$$

$$\text{If } f \not\equiv g,\ f(t_1, \ldots, t_n) = g(u_1, \ldots, u_m) \rightarrow \mathit{false} \tag{$=_6$}$$

$$\begin{array}{l}\{t_1, \ldots, t_m \sqcup \dot{A}\} = \{u_1, \ldots, u_n \sqcup \dot{A}\} \rightarrow \\ \quad t_1 = u_j \wedge \{t_2, \ldots, t_m \sqcup \dot{A}\} = \{u_1, \ldots, u_{j-1}, u_{j+1}, \ldots, u_n \sqcup \dot{A}\} \\ \quad \vee\, t_1 = u_j \wedge \{t_1, \ldots, t_m \sqcup \dot{A}\} = \{u_1, \ldots, u_{j-1}, u_{j+1}, \ldots, u_n \sqcup \dot{A}\} \\ \quad \vee\, t_1 = u_j \wedge \{t_2, \ldots, t_m \sqcup \dot{A}\} = \{u_1, \ldots, u_n \sqcup \dot{A}\} \\ \quad \vee\, \dot{A} = \{t_1 \sqcup N\} \wedge \{t_2, \ldots, t_m \sqcup N\} = \{u_1, \ldots, u_n \sqcup N\}\end{array} \tag{$=_7$}$$

$$\begin{array}{l}\{x \sqcup A\} = \{y \sqcup B\} \rightarrow \\ \quad x = y \wedge A = B \\ \quad \vee\, x = y \wedge \{x \sqcup A\} = B \\ \quad \vee\, x = y \wedge A = \{y \sqcup B\} \\ \quad \vee\, A = \{y \sqcup N\} \wedge \{x \sqcup N\} = B\end{array} \tag{$=_8$}$$

$$f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n) \rightarrow t_1 = u_1 \wedge \cdots \wedge t_n = u_n \tag{$=_9$}$$

Figure 4: Rewrite rules for =-constraints

Irreducible form:

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

3.2 Inequality

Syntax: $t_1 \neq t_2$.
Informal semantics: t_1 is different from t_2.
Rewrite rules: see Fig. 5.

If $t, t_i : \mathsf{U}$; $A, B : \mathsf{Set}$ then:

$$\dot{x} \neq \dot{x} \rightarrow \mathit{false} \qquad (\neq_1)$$

$$\text{If } t \notin \mathcal{V},\ t \neq \dot{x} \rightarrow \dot{x} \neq t \qquad (\neq_2)$$

$$\text{If } \dot{x} \notin \mathit{vars}(t_1, \ldots, t_n), \dot{x} \neq \{t_1, \ldots, t_n \sqcup \dot{x}\} \rightarrow \quad t_1 \notin \dot{x} \vee \cdots \vee t_n \notin \dot{x} \qquad (\neq_3)$$

$$\text{If } \dot{x} \in \mathit{vars}(t), \dot{x} \neq t \rightarrow \mathit{true} \qquad (\neq_4)$$

$$\{t_1 \sqcup A\} \neq \{t_2 \sqcup B\} \rightarrow \quad N \in \{t_1 \sqcup A\} \wedge N \notin \{t_2 \sqcup B\} \quad \vee N \notin \{t_1 \sqcup A\} \wedge N \in \{t_2 \sqcup B\} \qquad (\neq_5)$$

$$f(t_1, \ldots, t_n) \neq g(u_1, \ldots, u_n) \rightarrow \mathit{true} \qquad (\neq_6)$$

$$f(t_1, \ldots, t_n) \neq f(u_1, \ldots, u_n) \rightarrow t_1 \neq u_1 \vee \cdots \vee t_n \neq u_n \qquad (\neq_7)$$

Figure 5: 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 $p(\ldots)$, $p \in \{un, id, inv, comp\}$, 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. 6.

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

$$x \in \emptyset \to \mathit{false} \tag{$\in_1$}$$

$$x \in \{y \sqcup A\} \to x = y \vee x \in A \tag{$\in_2$}$$

$$x \in \dot{A} \to \dot{A} = \{x \sqcup N\} \tag{$\in_3$}$$

Figure 6: 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. 7.

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

$$un(\dot{A}, \dot{A}, B) \to \dot{A} = B \tag{$\cup_1$}$$

$$un(A, B, \emptyset) \to A = \emptyset \wedge B = \emptyset \tag{$\cup_2$}$$

$$un(\emptyset, A, \dot{B}) \to \dot{B} = A \tag{$\cup_3$}$$

$$un(A, \emptyset, \dot{B}) \to \dot{B} = A \tag{$\cup_4$}$$

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

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

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

Figure 7: Rewrite rules for *un*-constraints

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. 8.

If $t, t_i : \mathsf{U}$; $A, B : \mathsf{Set}$ then:

$$\emptyset \parallel A \to \mathit{true} \qquad (\parallel_1)$$

$$A \parallel \emptyset \to \mathit{true} \qquad (\parallel_2)$$

$$\dot{A} \parallel \dot{A} \to \dot{A} = \emptyset \qquad (\parallel_3)$$

$$\{t \sqcup B\} \parallel \dot{A} \to t \notin \dot{A} \wedge \dot{A} \parallel B \qquad (\parallel_4)$$

$$\dot{A} \parallel \{t \sqcup B\} \to t \notin \dot{A} \wedge \dot{A} \parallel B \qquad (\parallel_5)$$

$$\{t_1 \sqcup A\} \parallel \{t_2 \sqcup B\} \to t_1 \neq t_2 \wedge t_1 \notin B \wedge t_2 \notin A \wedge A \parallel B \qquad (\parallel_6)$$

Figure 8: Rewrite rules for $\parallel$-constraints (disjointness)

Irreducible form:

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

4.4 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. 9.

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

$$\dot{A} \subseteq \dot{A} \to \mathit{true} \tag{4.1}$$

$$\emptyset \subseteq A \to \mathit{true} \tag{4.2}$$

$$\dot{A} \subseteq \emptyset \to \dot{A} = \emptyset \tag{4.3}$$

$$\{x \sqcup A\} \subseteq \emptyset \to \mathit{false} \tag{4.4}$$

$$\{x \sqcup A\} \subseteq \dot{B} \to \quad \dot{B} = \{x \sqcup N\} \wedge A \subseteq \{x \sqcup N\} \tag{4.5}$$

$$\{x \sqcup A\} \subseteq \{y \sqcup B\} \to \quad x = y \wedge A \subseteq \{y \sqcup B\} \quad \vee x \neq y \wedge x \in B \wedge A \subseteq \{y \sqcup B\} \tag{4.6}$$

Figure 9: Rewrite rules for $\subseteq$-constraints

Irreducible forms:

- $\dot{A} \subseteq \dot{B}$, $\dot{A}$ and $\dot{B}$ distinct variables
- $\dot{A} \subseteq \{y|B\}$.

4.5 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. 10.

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

$$inters(\dot{A}, \dot{A}, B) \rightarrow A = B \tag{$\cup_8$}$$

$$inters(\emptyset, B, C) \rightarrow C = \emptyset \tag{$\cup_9$}$$

$$inters(A, \emptyset, C) \rightarrow C = \emptyset \tag{$\cup_{10}$}$$

$$inters(A, B, \emptyset) \rightarrow A \parallel B \tag{$\cup_{11}$}$$

$$\begin{aligned} &inters(A, B, \dot{C}) \rightarrow \\ &\quad A = \{x \sqcup N_1\} \\ &\quad \wedge B = \{x \sqcup N_2\} \wedge \dot{C} = \{x \sqcup N_3\} \wedge inters(N_1, N_2, N_3) \end{aligned} \tag{$\cup_{12}$}$$

$$\begin{aligned} &inters(A, B, \{x \sqcup C\}) \rightarrow \\ &\quad A = \{x \sqcup N_1\} \wedge B = \{x \sqcup N_2\} \wedge inters(N_1, N_2, C) \end{aligned} \tag{$\cup_{13}$}$$

Figure 10: Rewrite rules for *inters*-constraints

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

4.6 Difference

Syntax: $\mathit{diff}(t_1, t_2, t_3)$.
Informal semantics: if t_1, t_2 and t_3 are sets, then $t_3 = t_1 \setminus t_2$.
Rewrite rules: see Fig. 11.

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

$$\mathit{diff}(A, B, C) \rightarrow \mathit{un}(C, A, A) \wedge \mathit{un}(B, C, N) \wedge \mathit{un}(A, N, N) \tag{4.7}$$

Figure 11: 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. 12.

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

$$id(\dot{R}, \dot{R}) \to \dot{R} = \emptyset \tag{id_1}$$

$$id(\emptyset, R) \to R = \emptyset \tag{id_2}$$

$$id(A, \emptyset) \to A = \emptyset \tag{id_3}$$

$$id(\{a_1, \ldots, a_n \sqcup \dot{R}\}, \dot{R}) \to \mathit{false} \tag{id_4}$$

$$id(\dot{R}, \{(x_1, y_1), \ldots, (x_n, y_n) \sqcup \dot{R}\}) \to \mathit{false} \tag{id_5}$$

$$\begin{aligned} &id(\{a_1, \ldots, a_n \sqcup \dot{R}\}, \{(x_1, y_1), \ldots, (x_m, y_m) \sqcup \dot{R}\}) \to \\ &\quad id(\{a_1, \ldots, a_n\}, \{(x_1, y_1), \ldots, (x_m, y_m)\}) \wedge \dot{R} = \emptyset \end{aligned} \tag{id_6}$$

$$id(\{x \sqcup A\}, R) \to R = \{(x, x) \sqcup N\} \wedge id(A, N) \tag{id_7}$$

$$id(A, \{(x, y) \sqcup R\}) \to x = y \wedge A = \{x \sqcup N\} \wedge id(N, R) \tag{id_8}$$

Figure 12: 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. 13.

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

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

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

$$\begin{array}{l} inv(\dot{R}, \{(x_1, y_1), \ldots, (x_n, y_n) \sqcup \dot{R}\}) \to \\ \quad \dot{R} = \{(x_1, y_1), (y_1, x_1) \sqcup N\} \wedge inv(N, \{(x_2, y_2), \ldots, (x_n, y_n) \sqcup N\}) \end{array} \tag{3$^{\smile}$}$$

$$\begin{array}{l} inv(\{(x_1, y_1), \ldots, (x_n, y_n) \sqcup \dot{S}\}, \dot{S}) \to \\ \quad \dot{S} = \{(x_1, y_1), (y_1, x_1) \sqcup N\} \wedge inv(\{(x_2, y_2), \ldots, (x_n, y_n) \sqcup N\}, N) \end{array} \tag{4$^{\smile}$}$$

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

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

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

Figure 13: 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. 14.

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

$$comp(\emptyset, S, T) \to T = \emptyset \tag{$\circ_1$}$$

$$comp(R, \emptyset, T) \to T = \emptyset \tag{$\circ_2$}$$

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

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

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

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

Figure 14: 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. 15.

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

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

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

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

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

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

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

Figure 15: 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. 16.

If $R, A : \mathsf{Set}$; $x, y : \mathsf{U}$ 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 16: 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 $R, S, T, A, B, C, f : \mathsf{Set}$; $x, y : \mathsf{U}$ then:

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

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

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

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

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

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

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

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

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

Figure 17: 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 23 to 19.

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. 18.

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

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

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

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

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

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

Figure 18: 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. 19.

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

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

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

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

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

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

Figure 19: 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. 20.

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

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

Figure 20: Rewrite rules for *pair*-constraints

Irreducible form: none.

Sort constraint *set*

Syntax: $set(t)$.
Informal semantics: t is a set.
Rewrite rules: see Fig. 21.

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

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

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

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

Figure 21: 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. 22.

If $R : \mathsf{Set}$; $t : \mathsf{U}$ then:

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

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

Figure 22: 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. 23.

If $f : \mathsf{Set}; t : \mathsf{U}$ then:

$$pfun(\emptyset) \rightarrow true \qquad (\nrightarrow_1)$$

$$pfun(\{t \sqcup f\}) \rightarrow t = (n_1, n_2) \wedge comp(\{(n_1, n_1)\}, f, \emptyset) \wedge pfun(f) \qquad (\nrightarrow_2)$$

Figure 23: 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 24.

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

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

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

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

Figure 24: 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 25 and 24.

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

$$nset(\emptyset) \to \mathit{false} \qquad (\text{set}_4)$$

$$nset(\{t_1 \sqcup A\}) \to \mathit{false} \qquad (\text{set}_5)$$

$$nset(t_2) \to \mathit{true} \qquad (\text{set}_6)$$

Figure 25: 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. 31.

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

$$nrel(R) \to n \in R \wedge npair(n) \qquad (\leftrightarrow_3)$$

$$npfun(f) \to (n_1, n_2) \in f \wedge (n_1, n_3) \in f \wedge n_2 \neq n_3 \vee nrel(f) \qquad (\nrightarrow_3)$$

Figure 26: 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. 27.

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

$$x \notin \emptyset \to \mathit{true} \qquad (\in_4)$$
$$x \notin \{y \sqcup A\} \to x \neq y \wedge x \notin A \qquad (\in_5)$$
$$\text{If } \dot{A} \in \mathit{vars}(x), x \notin \dot{A} \to \mathit{true} \qquad (\in_6)$$

Figure 27: 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. 28.

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

$$nun(A, B, C) \rightarrow \\ N \in C \wedge N \notin A \wedge N \notin B \\ \vee\, N \in A \wedge N \notin C \\ \vee\, N \in B \wedge N \notin C \qquad (\cup_{14})$$

Figure 28: Rewrite rules for negative *un*-constraints

Irreducible form: none.

Not disjoint

Syntax: $t_1 \nparallel t_2$.
Informal semantics: if t_1 and t_2 are sets, then $t_1 \cap t_2 \neq \emptyset$.
Rewrite rules: see Fig. 29.

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

$$A \nparallel B \rightarrow n \in A \wedge n \in B \qquad (\|_7)$$

Figure 29: Rewrite rules for negative $\|$-constraints

Irreducible form: none.

7.3 Other negative set constraints

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

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

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

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

Figure 30: Rewrite rules for other negative set constraints

Irreducible form: none.

8 Rewrite rules for negative relational constraints

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

$$ndom(R, A) \rightarrow \\ (n_1, n_2) \in R \wedge n_1 \notin A \\ \vee\, n_1 \in A \wedge comp(\{(n_1, n_1)\}, R, \emptyset) \\ \vee\, nrel(R) \qquad (\mathrm{dom}_{18})$$

$$ninv(R, S) \rightarrow \\ (n_1, n_2) \in R \wedge (n_2, n_1) \notin S \\ \vee\, (n_1, n_2) \notin R \wedge (n_2, n_1) \in S \\ \vee\, nrel(R) \vee nrel(S) \qquad (8^{\smile})$$

$$ncomp(R, S, T) \rightarrow \\ (n_1, n_2) \in R \wedge (n_2, n_3) \in S \wedge (n_1, n_3) \notin T \\ \vee (n_1, n_3) \in T \\ \wedge\, comp(\{(n_1, n_1)\}, R, N_1) \\ \wedge\, comp(S, \{(n_3, n_3)\}, N_2) \wedge comp(N_1, N_2, \emptyset) \\ \vee\, nrel(R) \vee nrel(S) \vee nrel(T) \qquad (\circ_{12})$$

$$nid(A, f) \rightarrow \\ n_1 \in A \wedge (n_1, n_1) \notin f \\ \vee\, n_1 \notin A \wedge (n_1, n_1) \in f \\ \vee\, n_1 \neq n_2 \wedge (n_1, n_2) \in f \\ \vee\, n \in f \wedge npair(n) \qquad (\mathrm{id}_{13})$$

Figure 31: Rewrite rules for negative relational constraints

Irreducible form: none.

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

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

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

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

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

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

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

$$
\tag{8.7}
$$

Figure 32: Rewrite rules for negative relational constraints—cont'd

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

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

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

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

Figure 33: Rewrite rules for negative relational constraints—cont'd

9 The solver

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

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

```
Φ ← sort_infer(Φ);
repeat
    Φ′ ← Φ;
    repeat
        Φ″ ← Φ;
        Φ ← STEP(Φ)
    until Φ = Φ″;
    Φ ← remove_neq(Φ)
until Φ = Φ′;
return Φ
```

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 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 34.

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,
$\pi \in \{un, \subseteq, inters, id, inv, comp\}, X \neq t \rightarrow$ $\qquad (\neq_8)$
$\quad (n \in X \wedge n \notin t) \vee (n \in t \wedge n \notin X) \vee (X = \emptyset \wedge t \neq \emptyset)$

Figure 34: Rule scheme for $\neq$-constraint elimination

Remark 2 *The third disjunct in the rule scheme of Figure 34 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 $SAT_{\mathcal{BR}}$ is the procedure STEP. STEP is a function that takes as its input a $\mathcal{BR}$-formula Φ and returns a new $\mathcal{BR}$-formula Φ'. The overall structure of STEP is shown in Figure 35.

$$
\begin{array}{l}
\mathsf{STEP}(\Phi): \\
\quad \text{let } \Phi \text{ be } \Phi_1 \vee \cdots \vee \Phi_n,\ n \geq 1; \\
\quad \Phi \rightarrow \mathsf{STEP}_0(\Phi_1) \\
\qquad \vee \mathsf{STEP}_0(\Phi_2) \\
\qquad \ldots \\
\qquad \vee \mathsf{STEP}_0(\Phi_n);
\end{array}
\tag{9.9}
$$

$$
\begin{array}{l}
\mathsf{STEP}_0(\Phi): \\
\quad \text{for all } \pi \text{ in } \Pi\text{: } \Phi \leftarrow \mathsf{rw}_\pi(\Phi); \\
\quad \Phi \leftarrow \mathsf{sort\_check}(\mathsf{sort\_infer}(\Phi)); \\
\quad \Phi \leftarrow \mathsf{id\_cycle\_check\_all}(\Phi); \\
\quad \mathsf{return}\ \Phi;
\end{array}
\tag{9.10}
$$

Figure 35: The procedure STEP

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

Generic procedure rw_π

The procedure rw_π (see Figure 36) is a generic procedure, parametric with respect to the $\mathcal{BR}$-constraint predicate symbol π. For each $\pi \in \Pi$, rw_π implements a *rewriting procedure for* π. rw_π takes as its input a $\mathcal{BR}$-formula Φ and returns a new $\mathcal{BR}$-formula Φ' which is obtained from Φ by repeatedly applying to it all possible rewrite rules for π. A rewrite rule $C \rightarrow \Phi$ is *applicable* to a constraint D if D matches C. Applying an applicable rule $C \rightarrow \Phi$ to a constraint D occurring in a formula Ψ causes D to be replaced by Φ in Ψ.

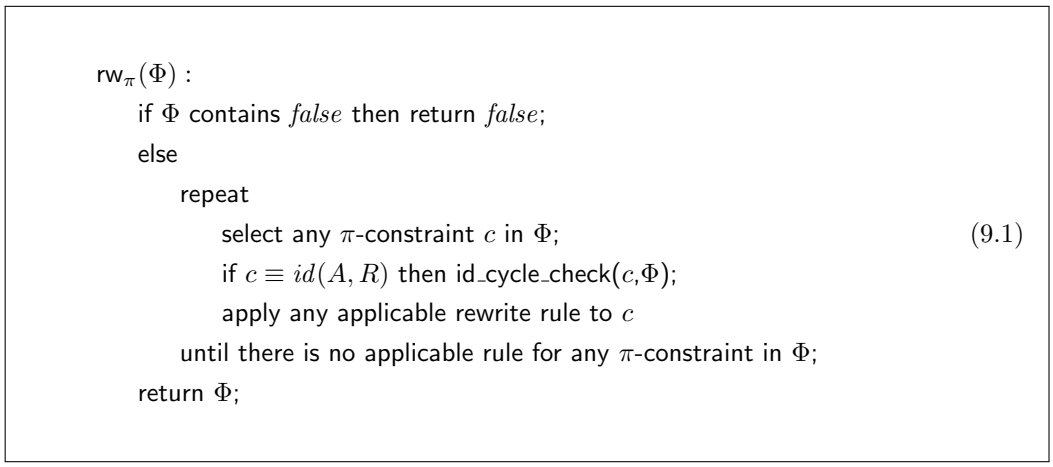

Figure 36: Rewriting procedure for π-constraints

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 37.

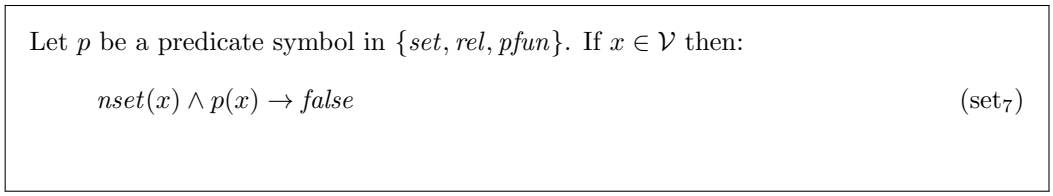

Figure 37: Rule scheme for sort consistency checking

Procedures id_cycle_check and id_cycle_check_all

The procedure id_cycle_check is shown in Fig. 38. It checks if Φ contains any *id* constraint whose first and second arguments share a variable, either directly (e.g., $id(\{X \sqcup R\}, R)$), or indirectly (e.g., $id(\{X \sqcup R\}, S) \wedge id(S, R)$)[1]. id_cycle_check uses a global graph structure, id_graph, represented as a set of unordered pairs $\langle x, y \rangle$ where x and y are variables occurring

[1]In the current implementation only direct sharing is detected by means of rules (id$_4$)–(id$_6$).

in Φ, and the following procedures: tail(t): returns the inner set part of the set term t, i.e., either $\emptyset$ or an unbound set variable x; add(g,p): adds the pair p to the set representing the graph g and returns the modified graph; cycle_occurs(g): checks whether the graph g contains a cycle and returns the set of all nodes participating in the cycle.

```
id_cycle_check(c, Φ) :
   let c be id(A, R);
   T_A = tail(A); T_R = tail(R);
   if T_A ∈ V ∧ T_R ∈ V then                                          (9.2)
       id_graph ← add(id_graph,⟨T_A, T_R⟩);
       NC ← cycle_occurs(id_graph);
       if NC ≠ ∅ then substitute any x ∈ NC by ∅ in all literals of Φ;
```

Figure 38: The procedure id_cycle_check

The procedure id_cycle_check_all checks if the formula obtained at the end of STEP contains any id constraint sharing variables between its arguments through some other constraints (e.g., $id(\{X \sqcup R\}, T) \wedge un(R, S, T)$). Note that initially id_graph contains all pairs resulting from the id constraints as detected by id_cycle_check. The procedure nodes(g) returns the set of all nodes in graph g.

```
id_cycle_check_all(Φ) :
    N ← nodes(id_graph);
    for all c in Φ:
        if c ≡ inv(R, S) then
            T_R = tail(R); T_S = tail(S);
            if T_R ∈ N ∨ T_S ∈ N then
                id_graph ← add(id_graph,⟨T_R, T_S⟩);
        if c ≡ un(R, S, T) ∨ c ≡ comp(R, S, T) then                    (9.3)
            T_R = tail(R); T_S = tail(S); T_T = tail(T);
            if T_R ∈ N ∧ T_T ∈ N then
                id_graph ← add(id_graph,⟨T_R, T_T⟩);
            if T_S ∈ N ∧ T_T ∈ N then
                id_graph ← add(id_graph,⟨T_S, T_T⟩);
    NC ← cycle_occurs(id_graph);
    if NC ≠ ∅ then substitute any x ∈ NC by ∅ in all literals of Φ;
```

Figure 39: The procedure id_cycle_check_all