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

Maximiliano Cristiá  Gianfranco Rossi
Universidad Nacional de Rosario  Università di Parma

January 9, 2019

Abstract

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

Contents

1. Conventions and notation 2
2. Sort inference rules 3
3. Rewrite rules for equality constraints 6
4. Rewrite rules for (positive) set constraints 8
5. Rewrite rules for (positive) relational constraints 14
6. Rewrite rules for sort constraints 22
7. Rewrite rules for negative set constraints 26
8. Rewrite rules for negative relational constraints 29
9. The solver 32
1 Conventions and notation

The set of primitive predicate symbols $\Pi$ is composed by the following collections of symbols:

- $\{=\}$ (equality constraints)
- $\{\in, \un, ||, \subseteq, \text{inters}, \text{diff}\}$ ((positive) set constraints)
- $\{\text{id}, \text{inv}, \text{comp}, \text{dom}, \text{ran}, \text{dres}, \text{dres}, \text{rres}, \text{rres}, \text{apply}, \text{rim}, \text{oplus}, \text{cp}\}$ ((positive) relational constraints)
- $\{\notin, \nun, \|, \text{nsubset}, \text{ninters}, \text{ndiff}\}$ (negative set constraints)
- $\{\text{nid}, \text{ninv}, \text{ncomp}, \text{ndom}, \text{nran}, \text{ndres}, \text{ndres}, \text{nrrres}, \text{nrrres}, \text{napply}, \text{nrimg}, \text{noplus}, \text{ncp}\}$ (negative relational constraints)
- $\{\text{pair}, \text{set}, \text{rel}, \text{pfun}, \text{npair}, \text{nset}, \text{nrel}, \text{npfun}\}$ (sort constraints).

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

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

$$\phi \rightarrow \Phi$$

where $\phi$ is a $\pi$-constraint and $\Phi$ is a $BR$-formula. If $\Phi$ has more than one disjunct then the rule is non-deterministic. Conjunctions occurring in $\Phi$ have higher precedence than disjunctions.

A rewriting procedure for $\pi$-constraints consists of the collection of all the rewrite rules for $\pi$-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 $\pi$-constraint $c$ is used to rewrite $c$. If no rules applies to $c$, then $c$ is left unchanged (i.e., $c$ is irreducible).

$L_{BR}$ defines also two sorts $\text{Set}$ and $\text{O}$ which, intuitively, represent the sort of set and non-set terms, respectively. For notational convenience, the following synonym is also defined: $U \equiv O \cup \text{Set}$.

Notational conventions.

- $V$ denotes a denumerable set of variables partitioned as $V \equiv V_{\text{Set}} \cup V_{\text{O}}$;
- 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 $x \in V$;
- $\text{vars}(t_1, \ldots, t_n)$ 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 \( t, u, v : U \) then:

\[
\begin{align*}
&\quad\quad t \in u \rightarrow t \in u \land set(u) \quad \text{(inf}_1) \\
&\quad\quad un(t, u, v) \rightarrow set(t) \land set(u) \land set(v) \quad \text{(inf}_2) \\
&\quad\quad t \parallel u \rightarrow set(t) \land set(u) \quad \text{(inf}_3) \\
&\quad\quad t \subseteq u \rightarrow t \subseteq u \land set(t) \land set(u) \quad \text{(inf}_4) \\
&\quad\quad inters(t, u, v) \rightarrow inters(t, u, v) \land set(t) \land set(u) \land set(v) \quad \text{(inf}_5) \\
&\quad\quad diff(t, u, v) \rightarrow diff(t, u, v) \land set(t) \land set(u) \land set(v) \quad \text{(inf}_6)
\end{align*}
\]

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

If \( t, u, v : U \) then:

\[
\begin{align*}
\text{inv}(t, u) & \rightarrow \text{inv}(t, u) \land \text{rel}(t) \land \text{rel}(u) \quad (\text{inf}_7) \\
\text{comp}(t, u, v) & \rightarrow \text{comp}(t, u, v) \land \text{rel}(t) \land \text{rel}(u) \land \text{rel}(v) \quad (\text{inf}_8) \\
\text{id}(t, u) & \rightarrow \text{id}(t, u) \land \text{set}(t) \land \text{rel}(u) \quad (\text{inf}_9) \\
\text{dom}(t, u) & \rightarrow \text{dom}(t, u) \land \text{rel}(t) \land \text{set}(u) \quad (\text{inf}_{10}) \\
\text{ran}(t, u) & \rightarrow \text{ran}(t, u) \land \text{rel}(t) \land \text{set}(u) \quad (\text{inf}_{11}) \\
\text{dres}(t, u, v) & \rightarrow \text{dres}(t, u, v) \land \text{set}(t) \land \text{rel}(u) \land \text{rel}(v) \quad (\text{inf}_{12}) \\
\text{rres}(t, u, v) & \rightarrow \text{rres}(t, u, v) \land \text{rel}(t) \land \text{set}(u) \land \text{rel}(v) \quad (\text{inf}_{13}) \\
\text{dares}(t, u, v) & \rightarrow \text{dares}(t, u, v) \land \text{set}(t) \land \text{rel}(u) \land \text{rel}(v) \quad (\text{inf}_{14}) \\
\text{rares}(t, u, v) & \rightarrow \text{rares}(t, u, v) \land \text{rel}(t) \land \text{set}(u) \land \text{rel}(v) \quad (\text{inf}_{15}) \\
\text{rim}(t, u, v) & \rightarrow \text{rim}(t, u, v) \land \text{rel}(t) \land \text{set}(u) \land \text{set}(v) \quad (\text{inf}_{16}) \\
\text{oplus}(t, u, v) & \rightarrow \text{oplus}(t, u, v) \land \text{rel}(t) \land \text{rel}(u) \land \text{rel}(v) \quad (\text{inf}_{17}) \\
\text{apply}(t, u, v) & \rightarrow \text{apply}(t, u, v) \land \text{rel}(t) \quad (\text{inf}_{18}) \\
\text{cp}(t, u, v) & \rightarrow \text{cp}(t, u, v) \land \text{set}(t) \land \text{set}(u) \land \text{rel}(v) \quad (\text{inf}_{19}) \\
\text{rel}(t) & \rightarrow \text{rel}(t) \land \text{set}(t) \quad (\text{inf}_{20}) \\
\text{pfun}(t) & \rightarrow \text{pfun}(t) \land \text{rel}(t) \quad (\text{inf}_{21})
\end{align*}
\]

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 \texttt{find_set} is used to find set terms, possibly occurring inside other terms, and to generate the corresponding set constraints. The definition of \texttt{find_set} is shown in Figure 3. We assume that all the \textit{true} constraints possibly generated by \texttt{find_set} are immediately removed via a trivial pre-processing.
find_set(t) :
    if \( t \equiv X \) or \( t \) is a constant symbol then return \( \text{true} \);
    if \( t \equiv f(t_1, \ldots, t_n), \ n > 0, \) and \( f \neq \{\cdot\} \)
        then return \( \text{find_set}(t_1) \land \cdots \land \text{find_set}(t_n) \);
    if \( t \equiv \{t_1, \ldots, t_n | t\} \)
        then return \( \text{find_set}(t_1) \land \cdots \land \text{find_set}(t_n) \land \text{set}(t) \); (2.1)

Figure 3: Finding set terms

Remark 1 \( \mathcal{L}_{BR} \) 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 \( \text{false} \). Ill-sorted literals are detected either by some rewrite rule (e.g., \( \text{un}(1, 2, \emptyset) \) is rewritten to \( \text{false} \) 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 \text{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 \( x \in 1 \land \text{set}(1) \) by rule \( (\text{inf}_1) \); in the further processing of this formula, literal \( x \in 1 \) is found to be irreducible since no rewrite rule for \( \varepsilon \)-constraints applies to it (see Fig. 6), while literal \( \text{set}(1) \) is rewritten to \( \text{false} \) by the rewrite rules for set-constraints (see Fig. 21); hence, the whole formula is rewritten to \( \text{false} \).
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 (\( \text{vars}(t_1, \ldots, t_n) \) denotes the set of variables occurring in \( t_1, \ldots, t_n \)).

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>If ( x, y, t, u_i : U; A, B : \text{Set} ) then: ( \hat{x} = \hat{x} \rightarrow \text{true} )</td>
</tr>
<tr>
<td>(2)</td>
<td>If ( t \notin V, t = \hat{x} \rightarrow \hat{x} = t )</td>
</tr>
<tr>
<td>(3)</td>
<td>If ( \hat{A} \notin \text{vars}(t_1, \ldots, t_n), \hat{A} = {t_1, \ldots, t_n \uplus \hat{A}} \rightarrow \hat{A} = {t_1, \ldots, t_n \uplus N} )</td>
</tr>
<tr>
<td>(4)</td>
<td>If ( \hat{x} \in \text{vars}(t), \hat{x} = t \rightarrow \text{false} )</td>
</tr>
<tr>
<td>(5)</td>
<td>If ( \hat{x} ) occurs in other literals of the input formula, ( \hat{x} = t \rightarrow \hat{x} = t ), and substitute ( \hat{x} ) by ( t ) in all other literals</td>
</tr>
<tr>
<td>(6)</td>
<td>If ( f \neq g, f(t_1, \ldots, t_n) = g(u_1, \ldots, u_m) \rightarrow \text{false} )</td>
</tr>
<tr>
<td>(7)</td>
<td>( {t_1, \ldots, t_m \uplus \hat{A}} = {u_1, \ldots, u_n \uplus \hat{A}} \rightarrow )</td>
</tr>
<tr>
<td></td>
<td>( t_1 = u_j \land {t_2, \ldots, t_m \uplus \hat{A}} = {u_1, \ldots, u_{j-1}, u_{j+1}, \ldots, u_n \uplus \hat{A}} )</td>
</tr>
<tr>
<td></td>
<td>( \lor t_1 = u_j \land {t_1, \ldots, t_m \uplus \hat{A}} = {u_1, \ldots, u_{j-1}, u_{j+1}, \ldots, u_n \uplus \hat{A}} )</td>
</tr>
<tr>
<td></td>
<td>( \lor t_1 = u_j \land {t_2, \ldots, t_m \uplus \hat{A}} = {u_1, \ldots, u_n \uplus \hat{A}} )</td>
</tr>
<tr>
<td></td>
<td>( \lor \hat{A} = {t_1 \uplus N} \land {t_2, \ldots, t_m \uplus N} = {u_1, \ldots, u_n \uplus N} )</td>
</tr>
<tr>
<td>(8)</td>
<td>( {x \uplus A} = {y \uplus B} \rightarrow )</td>
</tr>
<tr>
<td></td>
<td>( x = y \land A = B )</td>
</tr>
<tr>
<td></td>
<td>( \lor x = y \land {x \uplus A} = B )</td>
</tr>
<tr>
<td></td>
<td>( \lor x = y \land A = {y \uplus B} )</td>
</tr>
<tr>
<td></td>
<td>( \lor A = {y \uplus N} \land {x \uplus N} = B )</td>
</tr>
<tr>
<td></td>
<td>( f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n) \rightarrow t_1 = u_1 \land \cdots \land t_n = u_n )</td>
</tr>
</tbody>
</table>

Figure 4: Rewrite rules for =-constraints

Irreducible form:

- \( \hat{x} = t \) and neither \( t \) nor the other literals of the formula contain \( \hat{x} \).
3.2 Inequality

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

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\dot{x} \neq \dot{x} \rightarrow \text{false}$</td>
<td>($\neq_1$)</td>
</tr>
<tr>
<td>If $t \notin \mathcal{V}$, $t \neq \dot{x} \rightarrow \dot{x} \neq t$</td>
<td>($\neq_2$)</td>
</tr>
<tr>
<td>If $\dot{x} \notin \text{vars}(t_1, \ldots, t_n), \dot{x} \neq {t_1, \ldots, t_n \cup \dot{x}} \rightarrow$ $t_1 \notin \dot{x} \lor \cdots \lor t_n \notin \dot{x}$</td>
<td>($\neq_3$)</td>
</tr>
<tr>
<td>If $\dot{x} \in \text{vars}(t), \dot{x} \neq t \rightarrow \text{true}$</td>
<td>($\neq_4$)</td>
</tr>
<tr>
<td>${t_1 \cup A} \neq {t_2 \cup B} \rightarrow$ $N \in {t_1 \cup A} \land N \notin {t_2 \cup B}$ $\lor N \notin {t_1 \cup A} \land N \in {t_2 \cup B}$</td>
<td>($\neq_5$)</td>
</tr>
<tr>
<td>$f(t_1, \ldots, t_n) \neq g(u_1, \ldots, u_n) \rightarrow \text{true}$</td>
<td>($\neq_6$)</td>
</tr>
<tr>
<td>$f(t_1, \ldots, t_n) \neq f(u_1, \ldots, u_n) \rightarrow t_1 \neq u_1 \lor \cdots \lor t_n \neq u_n$</td>
<td>($\neq_7$)</td>
</tr>
</tbody>
</table>

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 $p(\ldots)$, $p \in \{\text{un, id, inv, comp}\}$, 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.

<table>
<thead>
<tr>
<th>If $x, y : U; A : Set$ then:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \in \emptyset \rightarrow false$</td>
</tr>
<tr>
<td>$x \in {y \cup A} \rightarrow x = y \lor x \in A$</td>
</tr>
<tr>
<td>$x \in \hat{A} \rightarrow \hat{A} = {x \cup N}$</td>
</tr>
</tbody>
</table>

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

**Irreducible form:** none.
4.2 Union

Syntax: \( \text{un}(t_1, t_2, t_3) \).

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 \( t : \text{U}; A, B, C : \text{Set} \) then:

\[
\begin{align*}
\text{un}(\dot{A}, \dot{A}, B) & \rightarrow \dot{A} = B & (\cup_1) \\
\text{un}(A, B, \emptyset) & \rightarrow A = \emptyset \land B = \emptyset & (\cup_2) \\
\text{un}(\emptyset, A, \dot{B}) & \rightarrow \dot{B} = A & (\cup_3) \\
\text{un}(A, \emptyset, \dot{B}) & \rightarrow \dot{B} = A & (\cup_4) \\
\text{un}(\{t \cup C\}, A, \dot{B}) & \rightarrow \\
& \land \{t \cup C\} = \{t \cup N_1\} \land \dot{B} = \{t \cup N\} & (\cup_5) \\
\text{un}(A, \{t \cup C\}, \dot{B}) & \rightarrow \\
& \land \{t \cup C\} = \{t \cup N_1\} \land \dot{B} = \{t \cup N\} & (\cup_6) \\
\text{un}(A, B, \{t \cup C\}) & \rightarrow \\
& \land \{t \cup C\} = \{t \cup N\} & (\cup_7)
\end{align*}
\]

Irreducible form:

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

Figure 7: Rewrite rules for \( \text{un} \)-constraints
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.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset \parallel A \rightarrow true )</td>
<td>(( \parallel_1 ))</td>
</tr>
<tr>
<td>( A \parallel \emptyset \rightarrow true )</td>
<td>(( \parallel_2 ))</td>
</tr>
<tr>
<td>( \hat{A} \parallel \hat{A} \rightarrow \hat{A} = \emptyset )</td>
<td>(( \parallel_3 ))</td>
</tr>
<tr>
<td>( {t \cup B} \parallel \hat{A} \rightarrow t \notin \hat{A} \land \hat{A} \parallel B )</td>
<td>(( \parallel_4 ))</td>
</tr>
<tr>
<td>( \hat{A} \parallel {t \cup B} \rightarrow t \notin \hat{A} \land \hat{A} \parallel B )</td>
<td>(( \parallel_5 ))</td>
</tr>
<tr>
<td>( {t_1 \cup A} \parallel {t_2 \cup B} \rightarrow t_1 \neq t_2 \land t_1 \notin B \land t_2 \notin A \land A \parallel B )</td>
<td>(( \parallel_6 ))</td>
</tr>
</tbody>
</table>

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

Irreducible form:

- \( \hat{A} \parallel \hat{B}, \hat{A} \) and \( \hat{B} \) 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 $x, y : U; A, B : \text{Set}$ then:

\begin{align*}
\hat{A} \subseteq \hat{A} & \rightarrow \text{true} \quad (4.1) \\
\emptyset \subseteq A & \rightarrow \text{true} \quad (4.2) \\
\hat{A} \subseteq \emptyset & \rightarrow \hat{A} = \emptyset \quad (4.3) \\
\{x \cup A\} \subseteq \emptyset & \rightarrow \text{false} \quad (4.4) \\
\{x \cup A\} \subseteq \hat{B} & \rightarrow \\
& \hat{B} = \{x \cup N\} \land A \subseteq \{x \cup N\} \\
\{x \cup A\} \subseteq \{y \cup B\} & \rightarrow \\
x = y \land A \subseteq \{y \cup B\} \\
\forall x \neq y \land x \in B \land A \subseteq \{y \cup B\} \quad (4.5)
\end{align*}

\begin{align*}
\{x \cup A\} \subseteq \{y \cup B\} & \rightarrow \\
x = y \land A \subseteq \{y \cup B\} \\
\forall x \neq y \land x \in B \land A \subseteq \{y \cup B\} \quad (4.6)
\end{align*}

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

Irreducible forms:

- $\hat{A} \subseteq \hat{B}$, $\hat{A}$ and $\hat{B}$ distinct variables
- $\hat{A} \subseteq \{y \mid B\}$. 

11
4.5 Intersection

Syntax: \( \text{inters}(t_1, t_2, t_3) \).

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.

\[
\begin{align*}
\text{If } x &: \text{U}; A, B, C &: \text{Set then:} \\
\text{inters}(A, A, B) &\rightarrow A = B \quad (\cup_8) \\
\text{inters}(\emptyset, B, C) &\rightarrow C = \emptyset \quad (\cup_9) \\
\text{inters}(A, \emptyset, C) &\rightarrow C = \emptyset \quad (\cup_{10}) \\
\text{inters}(A, B, \emptyset) &\rightarrow A \parallel B \quad (\cup_{11}) \\
\text{inters}(A, B, \hat{C}) &\rightarrow \\
\quad A = \{x \cup N_1\} \quad (\cup_{12}) \\
\quad \land B = \{x \cup N_2\} \land \hat{C} = \{x \cup N_3\} \land \text{inters}(N_1, N_2, N_3) \\
\text{inters}(A, B, \{x \cup C\}) &\rightarrow \\
\quad A = \{x \cup N_1\} \land B = \{x \cup N_2\} \land \text{inters}(N_1, N_2, C) \quad (\cup_{13}) \\
\end{align*}
\]

Figure 10: Rewrite rules for \textit{inters}-constraints

Irreducible forms:

- \( \text{inters}(\hat{A}, B, \hat{C}), \hat{A} \) and \( B \) are not the same variable
- \( \text{inters}(A, \hat{B}, \hat{C}), A \) and \( \hat{B} \) are not the same variable
4.6 Difference

*Syntax:* $\text{diff}(t_1, t_2, t_3)$.

*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.

<table>
<thead>
<tr>
<th>If $A, B, C : \text{Set}$ then:</th>
<th>diff($A, B, C$) $\rightarrow$ un($C, A, A$) $\land$ un($B, C, N$) $\land$ un($A, N, N$)</th>
</tr>
</thead>
</table>

Figure 11: Rewrite rules for $\text{diff}$-constraints

*Irreducible form:* none.
5 Rewrite rules for (positive) relational constraints

5.1 Identity

Syntax: id(t₁, t₂).

Informal semantics: if t₁ is a set and t₂ is a binary relation, then t₂ is the identity relation over set t₁.

Rewrite rules: see Fig. 12.

If R, A : Set; x, y, xᵢ, yᵢ, aᵢ : U, then:

\[
\begin{align*}
\text{id}(\hat{R}, \hat{R}) & \rightarrow \hat{R} = \emptyset \quad \text{(id₁)} \\
\text{id}(\emptyset, \hat{R}) & \rightarrow R = \emptyset \quad \text{(id₂)} \\
\text{id}(A, \emptyset) & \rightarrow A = \emptyset \quad \text{(id₃)} \\
\text{id}(\{a₁, \ldots, aₙ \uplus \hat{R}\}, \hat{R}) & \rightarrow \text{false} \quad \text{(id₄)} \\
\text{id}(\hat{R}, \{(x₁, y₁), \ldots, (xₙ, yₙ) \uplus \hat{R}\}) & \rightarrow \text{false} \quad \text{(id₅)} \\
\text{id}(\{a₁, \ldots, aₙ \uplus \hat{R}\}, \{(x₁, y₁), \ldots, (xₘ, yₘ) \uplus \hat{R}\}) & \rightarrow \text{id}(\{a₁, \ldots, aₙ\}, \{(x₁, y₁), \ldots, (xₘ, yₘ)\}) \land \hat{R} = \emptyset \quad \text{(id₆)} \\
\text{id}(\{x \uplus A\}, R) & \rightarrow R = \{(x, x) \uplus N\} \land \text{id}(A, N) \quad \text{(id₇)} \\
\text{id}(A, \{(x, y) \uplus R\}) & \rightarrow x = y \land A = \{x \uplus N\} \land \text{id}(N, R) \quad \text{(id₈)}
\end{align*}
\]

Figure 12: Rewrite rules for id-constraints

Irreducible form:

- id(A, R), A and R distinct variables.
5.2 Inverse

*Syntax:* inv(t₁, t₂).

*Informal semantics:* if t₁ and t₂ are binary relations, then t₂ = t₁⁻¹.

*Rewrite rules:* see Fig. 13.

If R, S : Set; x, y, xᵢ, yᵢ, aᵢ, bᵢ : U then:

\[
\begin{align*}
&\text{inv}(R, \emptyset) \rightarrow R = \emptyset \quad (1^-) \\
&\text{inv}(\emptyset, S) \rightarrow S = \emptyset \quad (2^-) \\
&\text{inv}(\hat{R}, \{(x_1, y_1), \ldots, (x_n, y_n) \cup \hat{R}\}) \rightarrow \\
&\hat{R} = \{(x_1, y_1), (y_1, x_1) \cup N\} \land \text{inv}(N, \{(x_2, y_2), \ldots, (x_n, y_n) \cup N\}) \quad (3^-) \\
&\text{inv}(\{(x_1, y_1), \ldots, (x_n, y_n) \cup \hat{S}\}, \hat{S}) \rightarrow \\
&\hat{S} = \{(x_1, y_1), (y_1, x_1) \cup N\} \land \text{inv}(\{(x_2, y_2), \ldots, (x_n, y_n) \cup N\}, N) \quad (4^-) \\
&\text{inv}(\{(x_1, y_1), \ldots, (x_n, y_n) \cup \hat{R}\}, \{(a_1, b_1), \ldots, (a_m, b_m) \cup \hat{R}\}) \rightarrow \\
&\{(y_1, x_1) \cup N_1\} = \{(a_1, b_1), \ldots, (a_m, b_m)\} \\
&\land \text{un}(\hat{R}, N_1, N_2) \land \text{inv}(\{(x_2, y_2), \ldots, (x_n, y_n) \cup \hat{R}\}, N_2) \\
&\lor (y_1, x_1) \notin \{(a_1, b_1), \ldots, (a_m, b_m)\} \land (x_1, y_1) \notin \{(a_1, b_1), \ldots, (a_m, b_m)\} \\
&\land \hat{R} = \{(x_1, y_1), (y_1, x_1) \cup N\} \land \{(y_1, x_1), \ldots, (x_n, y_n)\} \\
&\land \text{inv}(\{(x_2, y_2), \ldots, (x_n, y_n) \cup N\}, \{(a_1, b_1), \ldots, (a_m, b_m) \cup N\}) \\
&\lor \{(y_1, x_1) \cup N_3\} = \{(x_2, y_2), \ldots, (x_n, y_n)\} \land \text{un}(N, N_1, N_4) \\
&\land \text{inv}(N_4, \{(a_1, b_1), \ldots, (a_m, b_m) \cup N\}) \\
&\lor (y_1, x_1) \notin \{(a_1, b_1), \ldots, (a_m, b_m)\} \\
&\land \{(x_1, y_1) \cup N_5\} = \{(a_1, b_1), \ldots, (a_m, b_m)\} \\
&\land \hat{R} = \{(y_1, x_1) \cup N\} \land \text{un}(N, N_5, N_6) \\
&\land \text{inv}(\{(x_2, y_2), \ldots, (x_n, y_n) \cup N\}, N_6) \\
&\lor \text{inv}(R, \{(y, x) \cup S\}) \rightarrow R = \{(x, y) \cup N\} \land \text{inv}(N, S) \quad (6^-) \\
&\text{inv}(\{(x, y) \cup R\}, S) \rightarrow S = \{(y, x) \cup N\} \land \text{inv}(R, N) \quad (7^-)
\end{align*}
\]
5.3 Composition

Syntax: $\text{comp}(t_1, t_2, t_3)$.

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 $Q, R, S, T : \text{Set}; Q \neq \emptyset; t, u, x, z : U$ then:

\begin{align*}
\text{comp}(\emptyset, S, T) & \rightarrow T = \emptyset & (o_1) \\
\text{comp}(R, \emptyset, T) & \rightarrow T = \emptyset & (o_2) \\
\text{comp}((\{x, u\}, \{t, z\}, T)) & \rightarrow (u = t \land T = \{(x, z)\}) \lor (u \neq t \land T = \emptyset) & (o_3) \\
\text{comp}((\{x, u\} \cup R), \{(t, z) \cup S\}, \emptyset) & \rightarrow \\
& u \neq t \\
& \land \text{comp}((\{x, u\}, \emptyset, S) \land \text{comp}(R, \{(t, z)\}, \emptyset) \land \text{comp}(R, S, \emptyset) & (o_4) \\
\text{comp}((\{x, u\} \cup T), \{(u, z) \cup S\}, T) & \rightarrow \\
& \text{comp}((\{x, u\}), \{(u, z)\}, N_1) \\
& \land \text{comp}((\{x, u\}, S, N_2) \land \text{comp}(R, \{(t, z)\}, N_3) \\
& \land \text{comp}(R, S, N_4) \\
& \land \text{un}(N_1, N_2, N_3, N_4, T) & (o_5) \\
\text{comp}(R, S, \{(x, z) \cup T\}) & \rightarrow \\
& \text{un}(N_x, N_{rt}, R) \land \text{un}(N_z, N_{st}, S) \\
& N_x = \{(x, u) \cup N_1\} \land N_z = \{(u, z) \cup N_2\} \\
& \land \text{comp}((\{x, u\}, N_1, N_1) \land \text{comp}(N_2, \{(x, z)\}, N_2) \\
& \land \text{comp}(N_z, N_{st}, N_3) \land \text{comp}(N_{rt}, N_z, N_4) \land \text{comp}(N_{rt}, N_{st}, N_5) \\
& \land \text{un}(N_3, N_4, N_5, T) & (o_6) \\
\end{align*}

Figure 14: Rewrite rules for $\text{comp}$-constraints

Irreducible forms:

- $\text{comp}(R, S, T), S \neq \emptyset$
- $\text{comp}(R, \hat{S}, \hat{T}), R \neq \emptyset$
- $\text{comp}(\hat{R}, S, \emptyset)$
- $\text{comp}(R, \hat{S}, \emptyset)$
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 \(t_2 = \text{dom} t_1\).
Rewrite rules: see Fig. 15.

<table>
<thead>
<tr>
<th>If (R, A : \text{Set}; x, y : U) then:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{dom}(\hat{R}, \hat{R}) \rightarrow \hat{R} = \emptyset)</td>
<td>(dom₁)</td>
</tr>
<tr>
<td>(\text{dom}(R, \emptyset) \rightarrow R = \emptyset)</td>
<td>(dom₂)</td>
</tr>
<tr>
<td>(\text{dom}(\emptyset, A) \rightarrow A = \emptyset)</td>
<td>(dom₃)</td>
</tr>
<tr>
<td>(\text{dom}(\hat{R}, {x \cup A}) \rightarrow \text{un}(N_1, N_2, \hat{R}) \land \text{dom}(N_1, {x}) \land \text{dom}(N_2, A))</td>
<td>(dom₄)</td>
</tr>
<tr>
<td>(\text{dom}(\hat{R}, {x}) \rightarrow \text{comp}({(x, x)}, \hat{R}, \hat{R}) \land \hat{R} \neq \emptyset)</td>
<td>(dom₅)</td>
</tr>
<tr>
<td>(\text{dom}({(x, y) \cup R}, A) \rightarrow A = {x \cup N_1} \land \text{dom}(R, N_1))</td>
<td>(dom₆)</td>
</tr>
</tbody>
</table>

Figure 15: Rewrite rules for \(\text{dom}\)-constraints

Irreducible form:
- \(\text{dom} (\hat{R}, \hat{A}), \hat{R}\) and \(\hat{A}\) distinct variables.
5.5 Range

Syntax: \( \text{ran}(t_1, t_2) \).

Informal semantics: if \( t_1 \) is a binary relation and \( t_2 \) is a set, then \( t_2 = \text{ran} t_1 \).

Rewrite rules: see Fig. 16.

\[
\begin{align*}
\text{If } R, A : \text{Set}; x, y : U \text{ then:} \\
\text{ran}(\hat{R}, \hat{R}) & \rightarrow \hat{R} = \emptyset & (\text{ran}_7) \\
\text{ran}(R, \emptyset) & \rightarrow R = \emptyset & (\text{ran}_8) \\
\text{ran}(\emptyset, A) & \rightarrow A = \emptyset & (\text{ran}_9) \\
\text{ran}(\hat{R}, \{y \uplus A\}) & \rightarrow \text{un}(N_1, N_2, \hat{R}) \land \text{ran}(N_1, \{y\}) \land \text{ran}(N_2, A) & (\text{ran}_{10}) \\
\text{ran}(\hat{R}, \{y\}) & \rightarrow \text{comp}(\hat{R}, \{(y, y)\}, \hat{R}) \land \hat{R} \neq \emptyset & (\text{ran}_{11}) \\
\text{ran}(\{(x, y) \uplus R\}, A) & \rightarrow A = \{y \uplus N_1\} \land \text{ran}(R, N_1) & (\text{ran}_{12})
\end{align*}
\]

Figure 16: Rewrite rules for \( \text{ran} \)-constraints

Irreducible form:

- \( \text{ran}(\hat{R}, \hat{A}) \), \( \hat{R} \) and \( \hat{A} \) distinct variables.
5.6 Other relational constraints

The following are relational constraints that can be expressed as 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 BR-formula.

\[
\begin{align*}
\text{If } R, S, T, A, B, C, f : \text{Set}; & \quad x, y : U \text{ then:} \\
\text{ran}(R, A) \rightarrow \text{inv}(R, N) \land \text{dom}(N, A) & \quad (5.1) \\
dres(A, R, S) \rightarrow & \\
\text{un}(S_1, N_1, R) \land \text{dom}(S_2, N_2) \land N_2 \subseteq A \land \text{dom}(N_1, N_3) \land A \parallel N_3 & \quad (5.2) \\
rres(R, A, S) \rightarrow & \\
\text{un}(S_1, N_1, R) \land \text{ran}(S_2, N_2) \land N_2 \subseteq A \land \text{ran}(N_1, N_3) \land A \parallel N_3 & \quad (5.3) \\
dares(A, R, S) \rightarrow & \\
dres(A, R, T) \land \text{un}(S, T, R) \land S \parallel T & \quad (5.4) \\
rares(R, A, S) \rightarrow & \\
rres(R, A, T) \land \text{un}(S, T, R) \land S \parallel T & \quad (5.5) \\
rimg(R, A, B) \rightarrow \text{dres}(A, R, N) \land \text{ran}(N, B) & \quad (5.6) \\
aplus(R, S, T) \rightarrow \text{dom}(S, N_1) \land \text{dares}(N_1, R, N_2) \land \text{un}(N_2, S, T) & \quad (5.7) \\
apply(f, x, y) \rightarrow (x, y) \in f \land \text{pfun}(f) & \quad (5.8) \\
cp(A, B, R) \equiv & \\
\text{dom}(N_1, A) \land \text{ran}(N_1, N_2) \land N_2 \subseteq \{n\} \\
\land \text{dom}(N_2, B) \land \text{ran}(N_2, N_3) \land N_3 \subseteq \{n\} \\
\land \text{inv}(N_2, N_4) \land \text{comp}(N_1, N_4, R) & \quad (5.9)
\end{align*}
\]

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: \( \text{dom}(t_1, t_2) \).

Informal semantics: if \( t_1 \) is a partial function and \( t_2 \) is a set, then \( t_2 = \text{dom} t_1 \).

Rewrite rules: see Fig. 18.

<table>
<thead>
<tr>
<th>If ( f, A : \text{Set}; x, y : U ) then:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{dom}(\hat{f}, \hat{A}) \rightarrow \hat{f} = \emptyset )</td>
</tr>
<tr>
<td>( \text{dom}(f, \emptyset) \rightarrow f = \emptyset )</td>
</tr>
<tr>
<td>( \text{dom}(\emptyset, A) \rightarrow A = \emptyset )</td>
</tr>
<tr>
<td>( \text{dom}(\hat{f}, {x \cup A}) \rightarrow \hat{f} = {(x, n) \cup N} \land \text{dom}(N, A) )</td>
</tr>
<tr>
<td>( \text{dom}((x, y) \cup f), A) \rightarrow A = {x \cup N} \land \text{dom}(f, N) )</td>
</tr>
</tbody>
</table>

Figure 18: Rewrite rules for \( \text{dom} \)-constraints over partial functions

Irreducible form:

- \( \text{dom}(\hat{f}, \hat{A}) \), \( \hat{f} \) and \( \hat{A} \) distinct variables.
Composition of partial functions

Syntax: $\text{comp}(t_1, t_2, t_3)$.

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

\[
\begin{align*}
\text{comp}(\emptyset, g, h) & \rightarrow h = \emptyset & (\circ_7) \\
\text{comp}(f, \emptyset, h) & \rightarrow h = \emptyset & (\circ_8) \\
\text{comp}(f, g, \emptyset) & \rightarrow \text{ran}(f, N_1) \land \text{dom}(g, N_2) \land N_1 \parallel N_2 & (\circ_9) \\
\text{comp}((\{(x, y) \cup f\}, g, \hat{h})) & \rightarrow \\
& g = \{(y, n) \cup N_1\} \land \hat{h} = \{(x, n) \cup N_2\} \land \text{comp}(f, g, N_2) & (\circ_{10}) \\
& \lor \text{dom}(g, N_1) \land y \notin N_1 \land \text{comp}(f, g, \hat{h}) \\
\text{comp}(f, g, (\{(x, z) \cup h\})) & \rightarrow \\
& f = \{(x, n) \cup N_1\} \land g = \{(n, z) \cup N_2\} \land \text{comp}(N_1, g, h) & (\circ_{11})
\end{align*}
\]

Figure 19: Rewrite rules for $\text{comp}$-constraints over partial functions

Irreducible forms:

- $\text{comp}(f, g, \hat{h}), \ g \neq \emptyset$.
- $\text{comp}(f, \hat{g}, \hat{h}), \ f \neq \emptyset$. 

21
6 Rewrite rules for sort constraints

Sort constraint pair

Syntax: \( \operatorname{pair}(t) \).
Informal semantics: \( t \) is a pair.
Rewrite rules: see Fig. 20.

\[
\text{If } t : U \text{ then:} \\
\text{pair}(t) \rightarrow t = (n_1, n_2) \quad (\text{pair}_1)
\]

Figure 20: Rewrite rules for pair-constraints

Irreducible form: none.

Sort constraint set

Syntax: \( \operatorname{set}(t) \).
Informal semantics: \( t \) is a set.
Rewrite rules: see Fig. 21.

\[
\text{If } A : \text{Set}; t_1 : U; t_2 : O \text{ then:} \\
\text{set}(\emptyset) \rightarrow \text{true} \quad (\text{set}_1) \\
\text{set}(\{t_1 \cup A\}) \rightarrow \text{set}(A) \quad (\text{set}_2) \\
\text{set}(t_2) \rightarrow \text{false} \quad (\text{set}_3)
\]

Figure 21: Rewrite rules for set-constraints

Irreducible form:

\[ \bullet \text{set}(\hat{x}). \]
Sort constraint \( rel \)

Syntax: \( rel(t) \).

Informal semantics: if \( t \) is a set, then \( t \) is a binary relation.

Rewrite rules: see Fig. 22.

<table>
<thead>
<tr>
<th>If ( R : \text{Set}; t : \text{U} ) then:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( rel(\emptyset) \rightarrow \text{true} ) (( \leftrightarrow_1 ))</td>
</tr>
<tr>
<td>( rel({ t \cup R }) \rightarrow t = (n_1, n_2) \land rel(R) ) (( \leftrightarrow_2 ))</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>If ( f : \text{Set}; t : \text{U} ) then:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( pfun(\emptyset) \rightarrow \text{true} ) (( \rightarrow_1 ))</td>
</tr>
<tr>
<td>( pfun({ t \cup f }) \rightarrow t = (n_1, n_2) \land \text{comp}({ (n_1, n_1) }, f, \emptyset) \land pfun(f) ) (( \rightarrow_2 ))</td>
</tr>
</tbody>
</table>

Figure 23: Rewrite rules for \( pfun \)-constraints

Irreducible form:

- \( pfun(\dot{x}) \).
Sort constraint \textit{npair}

Syntax: \textit{npair}(t).

Informal semantics: \(t\) is not a pair.

Rewrite rules: see Figure 24.

\[
\begin{align*}
\text{If } t_1, \ldots, t_n, f(t_1, \ldots, t_n) &: U, n \geq 0 \text{ then:} \\
\text{npair}((t_1, t_2)) &\rightarrow \text{false} \quad \text{(pair}_2) \\
\text{If } f \not\equiv (\cdot, \cdot), \text{npair}(f(t_1, \ldots, t_n)) &\rightarrow \text{true} \quad \text{(pair}_3) \\
\text{If } n \not\equiv 2, \text{npair}(f(t_1, \ldots, t_n)) &\rightarrow \text{true} \quad \text{(pair}_4) 
\end{align*}
\]

Figure 24: Rewrite rules for negative \textit{pair}-constraints

Irreducible forms:

\begin{itemize}
\item \textit{npair}(\dddot{x}).
\end{itemize}

Sort constraint \textit{nset}

Syntax: \textit{nset}(t).

Informal semantics: \(t\) is not a set.

Rewrite rules: see Figures 25 and 24.

\[
\begin{align*}
\text{If } A &: \text{Set}; t_1 &: U; t_2 &: O \text{ then:} \\
\text{nset}(\emptyset) &\rightarrow \text{false} \quad \text{(set}_4) \\
\text{nset}\{t_1 \cup A\} &\rightarrow \text{false} \quad \text{(set}_5) \\
\text{nset}(t_2) &\rightarrow \text{true} \quad \text{(set}_6) 
\end{align*}
\]

Figure 25: Rewrite rules for negative \textit{set}-constraints

Irreducible forms:

\begin{itemize}
\item \textit{nset}(\dddot{x}).
\end{itemize}
Sort constraints \textit{nrel} and \textit{nfun}

**Syntax:** \textit{nrel}(t), \textit{nfun}(t).

*Informal semantics:* \textit{t} is not a relation, \textit{t} is not a partial function.

*Rewrite rules:* see Fig. 31.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nrel(R) $\rightarrow$ $n \in R \land npair(n)$</td>
<td>($\leftrightarrow_3$)</td>
</tr>
<tr>
<td>npfun(f) $\rightarrow$ $(n_1, n_2) \in f \land (n_1, n_3) \in f \land n_2 \neq n_3 \lor nrel(f)$</td>
<td>($\rightarrow_3$)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>If $x, y : U; A : Set$ then:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \notin \emptyset \rightarrow true$</td>
</tr>
<tr>
<td>$x \notin {y \cup A} \rightarrow x \neq y \land x \notin A$</td>
</tr>
<tr>
<td>If $\dot{A} \in \text{vars}(x), x \notin \dot{A} \rightarrow true$</td>
</tr>
</tbody>
</table>

Figure 27: Rewrite rules for $\notin$-constraints

Irreducible form:

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

26
7.2 Not union

Syntax: $\text{num}(t_1, t_2, t_3)$.
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 $A, B, C : \text{Set}$ then:
\[
\text{num}(A, B, C) \rightarrow \\
N \in C \land N \notin A \land N \notin B \\
\lor N \in A \land N \notin C \\
\lor N \in B \land N \notin C \\
\tag{\cup_{14}}
\]

Figure 28: Rewrite rules for negative $\text{un}$-constraints

Irreducible form: none.

Not disjoint

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

If $A, B : \text{Set}$ then:
\[
A \parallel B \rightarrow n \in A \land n \in B \\
\tag{\parallel_{7}}
\]

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

Irreducible form: none.
7.3 Other negative set constraints

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

- $\text{nsuset}(A, B) \rightarrow n \in A \land n \notin B$ \hspace{1cm} (7.1)

- $\text{ninters}(A, B, C) \rightarrow$
  \hspace{1cm} $n \in C \land (n \notin A \lor n \notin B)$
  \hspace{1cm} $\lor n \in A \land n \in B \land n \notin C$ \hspace{1cm} (7.2)

- $\text{ndiff}(A, B, C) \rightarrow$
  \hspace{1cm} $n \in C \land n \notin A \land n \notin B$
  \hspace{1cm} $\lor n \notin C \land n \in A \land n \notin B$ \hspace{1cm} (7.3)

Figure 30: Rewrite rules for other negative set constraints

Irreducible form: none.
8 Rewrite rules for negative relational constraints

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

\[
\begin{align*}
\text{ndom}(R, A) & \rightarrow \\
& (n_1, n_2) \in R \land n_1 \notin A \\
& \lor n_1 \in A \land \text{comp}([\{n_1, n_1\}], R, \emptyset) \\
& \lor \text{nrel}(R)
\end{align*}
\] (dom\textsubscript{18})

\[
\begin{align*}
\text{ninv}(R, S) & \rightarrow \\
& (n_1, n_2) \in R \land (n_2, n_1) \notin S \\
& \lor (n_1, n_2) \notin R \land (n_2, n_1) \in S \\
& \lor \text{nrel}(R) \lor \text{nrel}(S)
\end{align*}
\] (8\textsuperscript{–})

\[
\begin{align*}
\text{ncomp}(R, S, T) & \rightarrow \\
& (n_1, n_2) \in R \land (n_2, n_3) \in S \land (n_1, n_3) \notin T \\
& \lor (n_1, n_3) \in T \\
& \land \text{comp}([\{n_1, n_1\}], R, N_1) \\
& \land \text{comp}(S, \{n_3, n_3\}, N_2) \land \text{comp}(N_1, N_2, \emptyset) \\
& \lor \text{nrel}(R) \lor \text{nrel}(S) \lor \text{nrel}(T)
\end{align*}
\] (\circ\textsubscript{12})

\[
\begin{align*}
\text{nid}(A, f) & \rightarrow \\
n_1 \in A \land (n_1, n_1) \notin f \\
\lor n_1 \notin A \land (n_1, n_1) \in f \\
\lor n_1 \neq n_2 \land (n_1, n_2) \in f \\
\lor n \in f \land \text{npair}(n)
\end{align*}
\] (id\textsubscript{13})

Figure 31: Rewrite rules for negative relational constraints

\textbf{Irreducible form:} none.
If $R, S, T, A, B, C, f : \text{Set}$ then:

\[
nran(R, A) \rightarrow \\
(n_1, n_2) \in R \land n_2 \notin A \land n_1 \in A \land \text{comp}(R, \{(n_1, n_1)\}, \emptyset) \land \text{nrel}(R)
\]

(8.1)

\[
ndres(A, R, S) \rightarrow \\
(n_1, n_2) \in S \land n_1 \notin A \land n_1 \notin A \land (n_1, n_2) \notin R \land \text{nrel}(R) \lor \text{nrel}(S)
\]

(8.2)

\[
nrres(R, A, S) \rightarrow \\
(n_1, n_2) \in S \land n_2 \notin A \land n_2 \notin A \land (n_1, n_2) \notin R \land \text{nrel}(R) \lor \text{nrel}(S)
\]

(8.3)

\[
ndares(A, R, S) \rightarrow \\
(n_1, n_2) \in S \land n_1 \in A \land n_1 \notin A \land (n_1, n_2) \notin S \land \text{nrel}(R) \lor \text{nrel}(S)
\]

(8.4)

\[
nrares(R, A, S) \rightarrow \\
(n_1, n_2) \in S \land n_2 \in A \land n_2 \notin A \land (n_1, n_2) \notin S \land \text{nrel}(R) \lor \text{nrel}(S)
\]

(8.5)

\[
napply(f, x, y) \rightarrow (x, y) \notin f \lor \text{npfun}(f)
\]

(8.6)

\[
(8.7)
\]

Figure 32: Rewrite rules for negative relational constraints—cont’d
If \( R, S, T, A, B, C, f \): Set then:

\[
nrimg(R, A, B) \rightarrow \\
n_1 \in B \land \text{id}(A, N_1) \land \text{comp}(N_1, R, N_2) \land \text{comp}(N_2, \{(n_1, n_1)\}, \emptyset) \\
\lor n_1 \notin B \land (n_2, n_1) \in R \land n_2 \in A \\
\lor \text{nrel}(R)
\]  

\[8.8\]

\[
noplus(R, S, T) \rightarrow \\
(n_1, n_2) \in T \land (n_1, n_2) \notin S \land (n_1, n_2) \notin R \\
\lor (n_1, n_2) \in T \\
\land (n_1, n_2) \notin S \\
\land (n_1, n_2) \in R \land \text{comp}(\{(n_1, n_1)\}, S, \{(n_1, n_3) \cup N\}) \\
\lor (n_1, n_2) \notin T \land (n_1, n_2) \in S \\
\lor (n_1, n_2) \notin T \land (n_1, n_2) \in R \land \text{comp}(\{(n_1, n_1)\}, S, \emptyset) \\
\lor \text{nrel}(R) \lor \text{nrel}(S) \lor \text{nrel}(T)
\]  

\[8.9\]

\[
ncp(A, B, R) \rightarrow \\
n_1 \in A \land n_2 \in B \land (n_1, n_2) \notin R \\
\lor (n_1 \notin A \lor n_2 \notin B) \land (n_1, n_2) \in R
\]  

\[8.10\]

Figure 33: Rewrite rules for negative relational constraints—cont’d
9 The solver

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

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

\[
\begin{align*}
\Phi & \leftarrow \text{sort.infer}(\Phi); \\
& \text{repeat} \\
& \quad \Phi' \leftarrow \Phi; \\
& \quad \text{repeat} \\
& \quad \quad \Phi'' \leftarrow \Phi; \\
& \quad \quad \Phi \leftarrow \text{STEP}(\Phi) \\
& \quad \quad \text{until } \Phi = \Phi''; \\
& \quad \Phi \leftarrow \text{remove.neq}(\Phi) \\
& \quad \text{until } \Phi = \Phi'; \\
& \text{return } \Phi
\end{align*}
\]

Procedure sort.infer

The procedure sort.infer applies all possible sort inference rules (see Sect. 2) to all constraints occurring in the input formula $\Phi$.

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 $X \in V_{\text{Set}}; t : U; \Phi$ is the input formula then:

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

\[
\begin{align*}
\pi & \in \{ \text{un}, \subseteq, \text{inters}, \text{id}, \text{inv}, \text{comp} \}, X \neq t \rightarrow \\
(n \in X \land n \notin t) \lor (n \in t \land n \notin X) \lor (X = \emptyset \land t \neq \emptyset)
\end{align*}
\]

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\(_{BR}\) is the procedure **STEP**. **STEP** is a function that takes as its input a \(BR\)-formula \(\Phi\) and returns a new \(BR\)-formula \(\Phi'\). The overall structure of **STEP** is shown in Figure 35.

\[
\text{STEP}(\Phi): \\
\text{let } \Phi \text{ be } \Phi_1 \lor \cdots \lor \Phi_n, \ n \geq 1; \\
\Phi \rightarrow \text{STEP}_0(\Phi_1) \\
\lor \text{STEP}_0(\Phi_2) \\
\cdots \\
\lor \text{STEP}_0(\Phi_n); \\
\text{(9.9)}
\]

\[
\text{STEP}_0(\Phi): \\
\text{for all } \pi \text{ in } \Pi: \Phi \leftarrow \text{rw}_\pi(\Phi); \\
\Phi \leftarrow \text{sort.check(sort.infer(\Phi))}; \\
\Phi \leftarrow \text{id-cycle_check_all}(\Phi); \\
\text{return } \Phi; \\
\text{(9.10)}
\]

Figure 35: The procedure **STEP**

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

**Generic procedure** \(\text{rw}_\pi\)

The procedure \(\text{rw}_\pi\) (see Figure 36) is a generic procedure, parametric with respect to the \(BR\)-constraint predicate symbol \(\pi\). For each \(\pi \in \Pi\), \(\text{rw}_\pi\) implements a rewriting procedure for \(\pi\). \(\text{rw}_\pi\) takes as its input a \(BR\)-formula \(\Phi\) and returns a new \(BR\)-formula \(\Phi'\) which is obtained from \(\Phi\) by repeatedly applying to it all possible rewrite rules for \(\pi\). 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 \(\Psi\) causes \(D\) to be replaced by \(\Phi\) in \(\Psi\).
\[ \text{rw}_\pi(\Phi) : \]
\[ \text{if } \Phi \text{ contains } false \text{ then return } false; \]
\[ \text{else} \]
\[ \text{repeat} \]
\[ \text{select any } \pi\text{-constraint } c \text{ in } \Phi; \]
\[ \text{if } c \equiv id(A,R) \text{ then id\_cycle\_check}(c, \Phi); \]
\[ \text{apply any applicable rewrite rule to } c \]
\[ \text{until there is no applicable rule for any } \pi\text{-constraint in } \Phi; \]
\[ \text{return } \Phi; \]

Figure 36: Rewriting procedure for \(\pi\)-constraints

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

**Procedure sort\_check**

The procedure \text{sort\_check} applies the rewrite rules for sort checking to all possible pairs of sort constraints in \(\Phi\). If no pair exists for which the rules apply, then \(\Phi\) is returned unchanged. Otherwise \(\Phi\) is rewritten to \text{false}. The rewrite rules applied by \text{sort\_check} are described by the generic rule scheme of Figure 37.

\[ \text{Let } p \text{ be a predicate symbol in } \{\text{set, rel, pfun}\}. \text{ If } x \in \mathcal{V} \text{ then:} \]
\[ \text{nset}(x) \land p(x) \rightarrow false \]

Figure 37: Rule scheme for sort consistency checking

**Procedures id\_cycle\_check and id\_cycle\_check\_all**

The procedure \text{id\_cycle\_check} is shown in Fig. 38. It checks if \(\Phi\) contains any \text{id} constraint whose first and second arguments share a variable, either directly (e.g., \text{id}(\{X \cup R\}, R)), or indirectly (e.g., \text{id}(\{X \cup R\}, S) \land \text{id}(S, R))\(^1\). \text{id\_cycle\_check} uses a global graph structure, \text{id\_graph}, represented as a set of unordered pairs \((x, y)\) 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 $\Phi$, and the following procedures: $\text{tail}(t)$: returns the inner set part of the set term $t$, i.e., either $\emptyset$ or an unbound set variable $x$; $\text{add}(g,p)$: adds the pair $p$ to the set representing the graph $g$ and returns the modified graph; $\text{cycle\_occurs}(g)$: checks whether the graph $g$ contains a cycle and returns the set of all nodes participating in the cycle.

\[
\text{id\_cycle\_check}(c, \Phi) : \\
\text{let } c \text{ be } id(A,R); \\
T_A = \text{tail}(A); T_R = \text{tail}(R); \\
\text{if } T_A \in V \land T_R \in V \text{ then} \\
\quad \text{id\_graph} \leftarrow \text{add}(\text{id\_graph}, \langle T_A, T_R \rangle); \\
\quad NC \leftarrow \text{cycle\_occurs}(\text{id\_graph}); \\
\quad \text{if } NC \neq \emptyset \text{ then substitute any } x \in NC \text{ by } \emptyset \text{ in all literals of } \Phi; \\
\]

Figure 38: The procedure $\text{id\_cycle\_check}$

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

35
id_cycle_check_all(Φ) :
    N ← nodes(id_graph);
    for all c in Φ:
        if $c \equiv \text{inv}(R,S)$ then
            $T_R = \text{tail}(R); T_S = \text{tail}(S);$;
            if $T_R \in N \lor T_S \in N$ then
                id_graph $\leftarrow$ add(id_graph, ($T_R, T_S$));
        if $c \equiv \text{un}(R,S,T) \lor c \equiv \text{comp}(R,S,T)$ then
            $T_R = \text{tail}(R); T_S = \text{tail}(S); T_T = \text{tail}(T);$;
            if $T_R \in N \land T_T \in N$ then
                id_graph $\leftarrow$ add(id_graph, ($T_R, T_T$));
            if $T_S \in N \land T_T \in N$ then
                id_graph $\leftarrow$ add(id_graph, ($T_S, T_T$));
    NC ← cycle_occurs(id_graph);
    if $NC \neq \emptyset$ then substitute any $x \in NC$ by $\emptyset$ in all literals of $Φ$;

Figure 39: The procedure id_cycle_check_all