5.32 arith_or

DESCRIPTION LINKS GRAPH AUTOMATON

Origin Used in the definition of several automata

Constraint arith_or(VARIABLES1, VARIABLES2, RELOP, VALUE)

Arguments

VARIABLES1	:	collection(var−dvar)
VARIABLES2	:	collection(var−dvar)
RELOP	:	atom
VALUE	:	int

Restrictions

required(VARIABLES1, var)
required(VARIABLES2, var)
|VARIABLES1| = |VARIABLES2|
RELOP ∈ [=, ≠, <, ≥, >, ≤]

Purpose Enforce for all pairs of variables $\texttt{var1}_i, \texttt{var2}_i$ of the VARIABLES1 and VARIABLES2 collections to have $\texttt{var1}_i$ RELOP VALUE ∨ $\texttt{var2}_i$ RELOP VALUE.

Example $(\langle 0,1,0,0,1\rangle, \langle 0,0,0,1,0\rangle, =, 0)$

The constraint arith_or holds since, for all pairs of variables $\texttt{var1}_i, \texttt{var2}_i$ of the VARIABLES1 and VARIABLES2 collections, there is at least one variable that is equal to 0.

All solutions Figure 5.88 gives all solutions to the following non ground instance of the arith_or constraint: $\texttt{U}_1 \in [3,4]$, $\texttt{U}_2 \in [1,2]$, $\texttt{U}_3 \in [1,4]$, $\texttt{V}_1 \in [2,3]$, $\texttt{V}_2 \in [2,2]$, $\texttt{V}_3 \in [0,1]$, arith_or$(\langle \texttt{U}_1, \texttt{U}_2, \texttt{U}_3\rangle, \langle \texttt{V}_1, \texttt{V}_2, \texttt{V}_3\rangle, =, 2)$

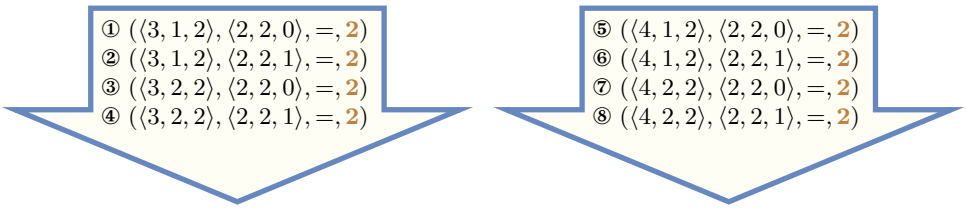


Figure 5.88: All solutions corresponding to the non ground example of the arith_or constraint of the **All solutions** slot

Typical

|VARIABLES1| > 0
RELOP ∈ [=]

Symmetries

- Arguments are permutable w.r.t. permutation (VARIABLES1, VARIABLES2) (RELOP) (VALUE).
- Items of VARIABLES1 and VARIABLES2 are permutable (*same permutation used*).

Arg. properties

Contractible wrt. `VARIABLES1` and `VARIABLES2` (*remove items from same position*).

See also

specialisation: `arith` (`variable RELOP VALUE` $\vee$ `variable RELOP VALUE` *replaced by* `variable RELOP VALUE`).

Keywords

characteristic of a constraint: automaton, automaton without counters, reified automaton constraint.

constraint network structure: Berge-acyclic constraint network.

constraint type: decomposition, value constraint.

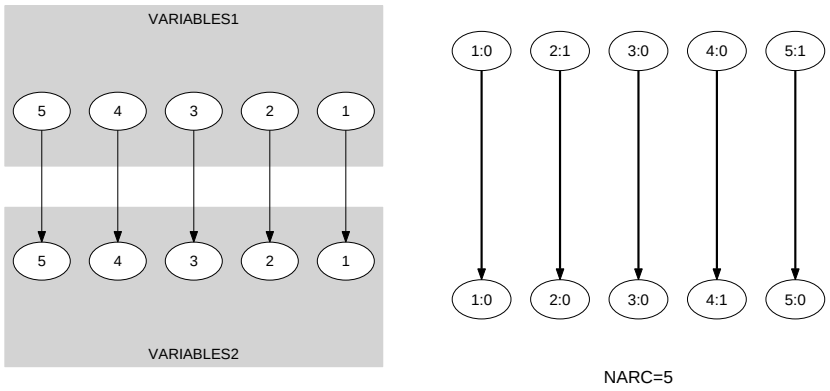
filtering: arc-consistency.

final graph structure: acyclic, bipartite, no loop.

modelling: disjunction.

Arc input(s)	VARIABLES1 VARIABLES2
Arc generator	$PRODUCT(=) \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	variables1.var RELOP VALUE $\vee$ variables2.var RELOP VALUE
Graph property(ies)	**NARC**$=$ |VARIABLES1|
Graph class	• ACYCLIC • BIPARTITE • NO_LOOP

Graph model Parts (A) and (B) of Figure 5.89 respectively show the initial and final graphs associated with the **Example** slot. Since we use the **NARC** graph property, the arcs of the final graph are stressed in bold.



(A) **(B)**

Figure 5.89: Initial and final graph of the arith_or constraint

Automaton

Figure 5.90 depicts the automaton associated with the `arith_or` constraint. Let $\texttt{VAR1}_i$ and $\texttt{VAR2}_i$ be the i^{th} variables of the `VARIABLES1` and `VARIABLES2` collections. To each pair of variables $(\texttt{VAR1}_i, \texttt{VAR2}_i)$ corresponds a signature variable S_i. The following signature constraint links $\texttt{VAR1}_i$, $\texttt{VAR2}_i$ and S_i: $\texttt{VAR1}_i$ `RELOP VALUE` $\vee$ $\texttt{VAR2}_i$ `RELOP VALUE` $\Leftrightarrow S_i$. The automaton enforces for each pair of variables $\texttt{VAR1}_i$,$\texttt{VAR2}_i$ the condition $\texttt{VAR1}_i$ `RELOP VALUE` $\vee$ $\texttt{VAR2}_i$ `RELOP VALUE`.

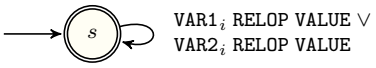


Figure 5.90: Automaton of the `arith_or` constraint

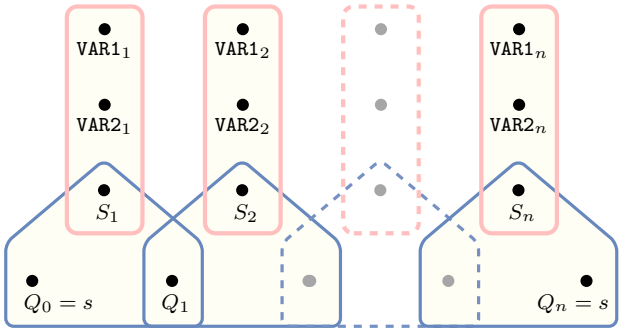


Figure 5.91: Hypergraph of the reformulation corresponding to the automaton of the `arith_or` constraint