5.105 disjoint

DESCRIPTION LINKS GRAPH AUTOMATON

Origin Derived from alldifferent.

Constraint disjoint(VARIABLES1, VARIABLES2)

Arguments

VARIABLES1	:	collection(var−dvar)
VARIABLES2	:	collection(var−dvar)

Restrictions

required(VARIABLES1, var)
required(VARIABLES2, var)

Purpose Each variable of the collection VARIABLES1 should take a value that is distinct from all the values assigned to the variables of the collection VARIABLES2.

Example

$$\left\{ \begin{array}{l} \langle 1, 9, 1, 5 \rangle, \\ \left\langle \begin{array}{l} \mathtt{var} - 2, \\ \mathtt{var} - 7, \\ \mathtt{var} - 7, \\ \mathtt{var} - 0, \\ \mathtt{var} - 6, \\ \mathtt{var} - 8 \end{array} \right\rangle \end{array} \right\}$$

In this example, values $1, 5, 9$ are used by the variables of VARIABLES1 and values $0, 2, 6, 7, 8$ by the variables of VARIABLES2. Since there is no intersection between the two previous sets of values the disjoint constraint holds.

Typical

$|\mathtt{VARIABLES1}| > 1$
$|\mathtt{VARIABLES2}| > 1$

Symmetries

- Arguments are permutable w.r.t. permutation (VARIABLES1, VARIABLES2).
- Items of VARIABLES1 are permutable.
- Items of VARIABLES2 are permutable.
- An occurrence of a value of VARIABLES1.var can be replaced by any value of VARIABLES1.var.
- An occurrence of a value of VARIABLES2.var can be replaced by any value of VARIABLES2.var.
- All occurrences of two distinct values in VARIABLES1.var or VARIABLES2.var can be swapped; all occurrences of a value in VARIABLES1.var or VARIABLES2.var can be renamed to any unused value.

Remark Despite the fact that this is not an uncommon constraint, it can not be modelled in a compact way neither with a *disequality* constraint (i.e., two given variables have to take distinct

values) nor with the alldifferent constraint. The disjoint constraint can bee seen as a special case of the common(NCOMMON1, NCOMMON2, VARIABLES1, VARIABLES2) constraint where NCOMMON1 and NCOMMON2 are both set to 0.

Algorithm

Let us note:

- n_1 the minimum number of distinct values taken by the variables of the collection VARIABLES1.
- n_2 the minimum number of distinct values taken by the variables of the collection VARIABLES2.
- n_{12} the maximum number of distinct values taken by the union of the variables of VARIABLES1 and VARIABLES2.

One invariant to maintain for the disjoint constraint is $n_1 + n_2 \leq n_{12}$. A lower bound of n_1 and n_2 can be obtained by using the algorithms provided in [26, 36]. An exact upper bound of n_{12} can be computed by using a bipartite matching algorithm.

Used in

k_disjoint.

See also

generalisation: disjoint_tasks *(*variable *replaced by* task*)*.

implies: alldifferent_on_intersection, lex_different.

system of constraints: k_disjoint.

Keywords

characteristic of a constraint: disequality, automaton, automaton with array of counters.

constraint type: value constraint.

filtering: bipartite matching.

modelling: empty intersection.

Arc input(s)	VARIABLES1 VARIABLES2
Arc generator	$PRODUCT \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	variables1.var = variables2.var
Graph property(ies)	NARC = 0

Graph model $PRODUCT$ is used in order to generate the arcs of the graph between all variables of VARIABLES1 and all variables of VARIABLES2. Since we use the graph property NARC = 0 the final graph will be empty. Figure 5.215 shows the initial graph associated with the **Example** slot. Since we use the NARC = 0 graph property the final graph is empty.

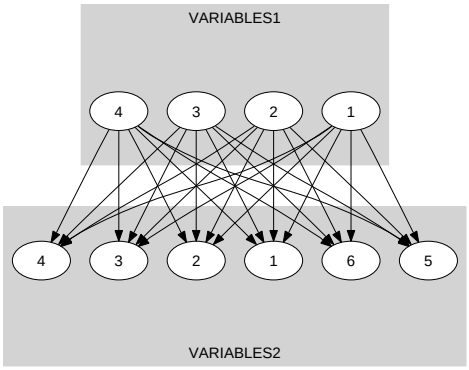

Figure 5.215: Initial graph of the disjoint constraint (the final graph is empty)

Signature Since 0 is the smallest number of arcs of the final graph we can rewrite NARC = 0 to NARC ≤ 0. This leads to simplify NARC to NARC.

Automaton

Figure 5.216 depicts the automaton associated with the `disjoint` constraint. To each variable $\texttt{VAR1}_i$ of the collection `VARIABLES1` corresponds a signature variable $\texttt{S}_i$ that is equal to 0. To each variable $\texttt{VAR2}_i$ of the collection `VARIABLES2` corresponds a signature variable $\texttt{S}_{i+|\texttt{VARIABLES1}|}$ that is equal to 1.

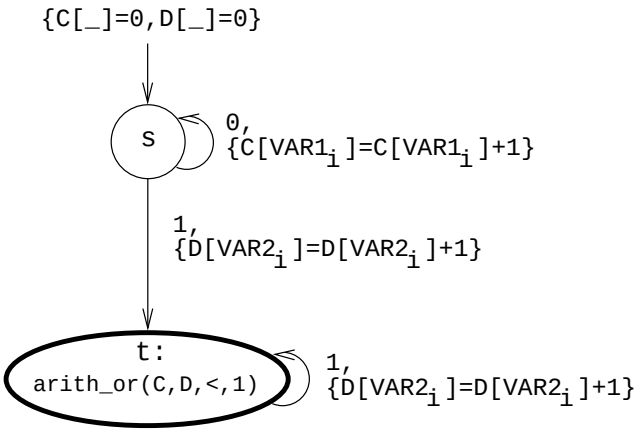

Figure 5.216: Automaton of the `disjoint` constraint