5.399 symmetric_gcc

DESCRIPTION LINKS GRAPH

Origin Derived from global_cardinality by W. Kocjan.

Constraint symmetric_gcc(VARS, VALS)

Synonym sgcc.

Arguments

$$\begin{array}{lll} \text{VARS} & : & \text{collection}(\text{idvar}-\text{int}, \text{var}-\text{svar}, \text{nocc}-\text{dvar}) \\ \text{VALS} & : & \text{collection}(\text{idval}-\text{int}, \text{val}-\text{svar}, \text{nocc}-\text{dvar}) \end{array}$$

Restrictions

$\text{required}(\text{VARS}, [\text{idvar}, \text{var}, \text{nocc}])$
$|\text{VARS}| \geq 1$
$\text{VARS.idvar} \geq 1$
$\text{VARS.idvar} \leq |\text{VARS}|$
$\text{distinct}(\text{VARS}, \text{idvar})$
$\text{VARS.nocc} \geq 0$
$\text{VARS.nocc} \leq |\text{VALS}|$
$\text{required}(\text{VALS}, [\text{idval}, \text{val}, \text{nocc}])$
$|\text{VALS}| \geq 1$
$\text{VALS.idval} \geq 1$
$\text{VALS.idval} \leq |\text{VALS}|$
$\text{distinct}(\text{VALS}, \text{idval})$
$\text{VALS.nocc} \geq 0$
$\text{VALS.nocc} \leq |\text{VARS}|$

Purpose Put in relation two sets: for each element of one set gives the corresponding elements of the other set to which it is associated. In addition, enforce a cardinality constraint on the number of occurrences of each value.

Example

$$\left(\begin{array}{l} \left\langle\begin{array}{lll} \text{idvar}-1 & \text{var}-\{3\} & \text{nocc}-1, \\ \text{idvar}-2 & \text{var}-\{1\} & \text{nocc}-1, \\ \text{idvar}-3 & \text{var}-\{1,2\} & \text{nocc}-2, \\ \text{idvar}-4 & \text{var}-\{1,3\} & \text{nocc}-2 \end{array}\right\rangle, \\ \left\langle\begin{array}{lll} \text{idval}-1 & \text{val}-\{2,3,4\} & \text{nocc}-3, \\ \text{idval}-2 & \text{val}-\{3\} & \text{nocc}-1, \\ \text{idval}-3 & \text{val}-\{1,4\} & \text{nocc}-2, \\ \text{idval}-4 & \text{val}-\emptyset & \text{nocc}-0 \end{array}\right\rangle \end{array}\right)$$

The symmetric_gcc constraint holds since:

- $3 \in \text{VARS}[1].\text{var} \Leftrightarrow 1 \in \text{VALS}[3].\text{val}$,
- $1 \in \text{VARS}[2].\text{var} \Leftrightarrow 2 \in \text{VALS}[1].\text{val}$,
- $1 \in \text{VARS}[3].\text{var} \Leftrightarrow 3 \in \text{VALS}[1].\text{val}$,
- $2 \in \text{VARS}[3].\text{var} \Leftrightarrow 3 \in \text{VALS}[2].\text{val}$,

- $1 \in$ VARS$[4]$.var $\Leftrightarrow 4 \in$ VALS$[1]$.val,
- $3 \in$ VARS$[4]$.var $\Leftrightarrow 4 \in$ VALS$[3]$.val,
- The number of elements of VARS$[1]$.var $= \{3\}$ is equal to 1,
- The number of elements of VARS$[2]$.var $= \{1\}$ is equal to 1,
- The number of elements of VARS$[3]$.var $= \{1, 2\}$ is equal to 2,
- The number of elements of VARS$[4]$.var $= \{1, 3\}$ is equal to 2,
- The number of elements of VALS$[1]$.val $= \{2, 3, 4\}$ is equal to 3,
- The number of elements of VALS$[2]$.val $= \{3\}$ is equal to 1,
- The number of elements of VALS$[3]$.val $= \{1, 4\}$ is equal to 2,
- The number of elements of VALS$[4]$.val $= \emptyset$ is equal to 0.

Typical

$|\text{VARS}| > 1$
$|\text{VALS}| > 1$

Symmetries

- Items of VARS are permutable.
- Items of VALS are permutable.

Usage

The most simple example of applying symmetric_gcc is a variant of personnel assignment problem, where one person can be assigned to perform between n and m ($n \leq m$) jobs, and every job requires between p and q ($p \leq q$) persons. In addition every job requires different kind of skills. The previous problem can be modelled as follows:

- For each person we create an item of the VARS collection,
- For each job we create an item of the VALS collection,
- There is an arc between a person and the particular job if this person is qualified to perform it.

Remark

The symmetric_gcc constraint generalises the global_cardinality constraint by allowing a variable to take more than one value. It corresponds to a variant of the symmetric_cardinality constraint described in [241] where the occurrence variables of the VARS and VALS collections are replaced by fixed intervals.

See also

common keyword: link_set_to_booleans *(constraint involving set variables)*.

root concept: global_cardinality.

specialisation: symmetric_cardinality (variable *replaced by* fixed interval).

used in graph description: in_set.

Keywords

application area: assignment.

combinatorial object: relation.

constraint arguments: constraint involving set variables.

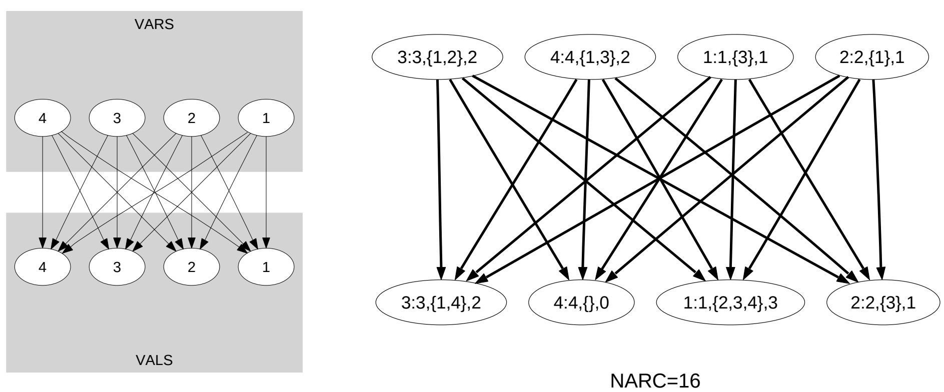
constraint type: decomposition, timetabling constraint.

filtering: flow.

Arc input(s)	VARS VALS
Arc generator	$PRODUCT \mapsto$ `collection`(`vars`, `vals`)
Arc arity	2
Arc constraint(s)	• `in_set`(`vars.idvar`, `vals.val`) $\Leftrightarrow$ `in_set`(`vals.idval`, `vars.var`) • `vars.nocc` = `card_set`(`vars.var`) • `vals.nocc` = `card_set`(`vals.val`)
Graph property(ies)	$\mathbf{NARC} = \lvert \mathtt{VARS} \rvert * \lvert \mathtt{VALS} \rvert$

Graph model The graph model used for the `symmetric_gcc` is similar to the one used in the `domain_constraint` or in the `link_set_to_booleans` constraints: we use an equivalence in the arc constraint and ask all arc constraints to hold.

Parts (A) and (B) of Figure 5.757 respectively show the initial and final graph. Since we use the **NARC** graph property, all the arcs of the final graph are stressed in bold.



(A) **(B)**

Figure 5.757: Initial and final graph of the `symmetric_gcc` constraint

Signature Since we use the *PRODUCT* arc generator on the collections VARS and VALS, the number of arcs of the initial graph is equal to $\lvert \mathtt{VARS} \rvert \cdot \lvert \mathtt{VALS} \rvert$. Therefore the maximum number of arcs of the final graph is also equal to $\lvert \mathtt{VARS} \rvert \cdot \lvert \mathtt{VALS} \rvert$ and we can rewrite $\mathbf{NARC} = \lvert \mathtt{VARS} \rvert \cdot \lvert \mathtt{VALS} \rvert$ to $\mathbf{NARC} \geq \lvert \mathtt{VARS} \rvert \cdot \lvert \mathtt{VALS} \rvert$. So we can simplify $\underline{\overline{\mathbf{NARC}}}$ to $\overline{\mathbf{NARC}}$.