5.299 open_global_cardinality

DESCRIPTION LINKS GRAPH

Origin [427]

Constraint open_global_cardinality(S, VARIABLES, VALUES)

Synonyms open_gcc, ogcc.

Arguments

S	:	svar
VARIABLES	:	collection(var−dvar)
VALUES	:	collection(val−int, noccurrence−dvar)

Restrictions

$\texttt{S} \geq 1$
$\texttt{S} \leq |\texttt{VARIABLES}|$
required(VARIABLES, var)
required(VALUES, [val, noccurrence])
distinct(VALUES, val)
$\texttt{VALUES.noccurrence} \geq 0$
$\texttt{VALUES.noccurrence} \leq |\texttt{VARIABLES}|$

Purpose

Each value VALUES$[i]$.val ($1 \leq i \leq |\texttt{VALUES}|$) should be taken by exactly VALUES$[i]$.noccurrence variables of the VARIABLES collection for which the corresponding position belongs to the set S. Positions are numbered from 1.

Example

$$\left(\begin{array}{l}\{2,3,4\},\\ \langle 3,3,8,6\rangle,\\ \left\langle\begin{array}{ll}\texttt{val}-3 & \texttt{noccurrence}-1,\\ \texttt{val}-5 & \texttt{noccurrence}-0,\\ \texttt{val}-6 & \texttt{noccurrence}-1\end{array}\right\rangle\end{array}\right)$$

The open_global_cardinality constraint holds since:

- Values 3, 5 and 6 respectively occur 1, 0 and 1 times within the collection $\langle 3,3,8,6\rangle$ (the first item 3 of $\langle 3,3,8,6\rangle$ is ignored since value 1 does not belong to the first argument $S = \{2,3,4\}$ of the open_global_cardinality constraint).
- No constraint was specified for value 8.

Typical

$|\texttt{VARIABLES}| > 1$
$\text{range}(\texttt{VARIABLES.var}) > 1$
$|\texttt{VALUES}| > 1$
$\text{range}(\texttt{VALUES.noccurrence}) > 1$
$|\texttt{VARIABLES}| > |\texttt{VALUES}|$

Symmetries

- Items of VALUES are permutable.
- An occurrence of a value of VARIABLES.var that does not belong to VALUES.val can be replaced by any other value that also does not belong to VALUES.val.

Usage In their article [427], W.-J. van Hoeve and J.-C. Régin motivate the `open_global_cardinality` constraint by the following scheduling problem. Consider a set of activities (where each activity has a fixed duration 1 and a start variable) that can be processed on two factory lines such that all the activities that will be processed on a given line must be pairwise distinct. This can be modelled by using one `open_global_cardinality` constraint for each line, involving all the start variables as well as a set variable whose final value specifies the set of activities assigned to that specific factory line.

Note that this can also be directly modelled by a single `diffn` constraint. This is done by introducing an assignment variable for each activity. The initial domain of each assignment variable consists of two values that respectively correspond to the two factory lines.

Remark In their article [427], W.-J. van Hoeve and J.-C. Régin consider the case where we have no counter variables for the values, but rather some lower and upper bounds (i.e., in fact the `open_global_cardinality_low_up` constraint).

Algorithm A slight adaptation of the flow model that handles the original `global_cardinality` constraint [342] is described in [427].

See also

common keyword: `global_cardinality_low_up` *(assignment,counting constraint)*, `open_among` *(open constraint,counting constraint)*, `open_atleast`, `open_atmost` *(open constraint,value constraint)*.

hard version: `global_cardinality`.

specialisation: `open_alldifferent` *(each active value[14] should occur at most once)*, `open_global_cardinality_low_up` *(*`variable` *replaced by* `fixed interval`*)*.

used in graph description: `in_set`.

Keywords

application area: assignment.

constraint arguments: constraint involving set variables.

constraint type: open constraint, value constraint, counting constraint.

filtering: flow.

[14]An *active value* corresponds to a value occuring at a position mentionned in the set `S`.

	For all items of `VALUES`:
Arc input(s)	`VARIABLES`
Arc generator	*SELF* $\mapsto$ `collection`(`variables`)
Arc arity	1
Arc constraint(s)	• `variables.var` = `VALUES.val` • `in_set`(`variables.key`, `S`)
Graph property(ies)	**NVERTEX** = `VALUES.noccurrence`

Graph model Since we want to express one unary constraint for each value we use the "For all items of `VALUES`" iterator. The only difference with the graph model of the `global_cardinality` constraint is the arc constraint where we also specify that the position of the considered variable should belong to the first argument `S`.

Part (A) of Figure 5.634 shows the initial graphs associated with each value 3, 5 and 6 of the `VALUES` collection of the **Example** slot. Part (B) of Figure 5.634 shows the two corresponding final graphs respectively associated with values 3 and 6 that are both assigned to those variables of the `VARIABLES` collection for which the index belongs to `S` (since value 5 is not assigned to any variable of the `VARIABLES` collection the final graph associated with value 5 is empty). Since we use the **NVERTEX** graph property, the vertices of the final graphs are stressed in bold.

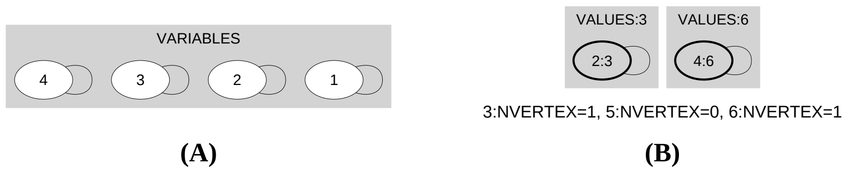

Figure 5.634: Initial and final graph of the `open_global_cardinality` constraint