5.78 counts

DESCRIPTION LINKS GRAPH AUTOMATON

Origin Derived from count.

Constraint counts(VALUES, VARIABLES, RELOP, LIMIT)

Arguments

VALUES	:	collection(val−int)
VARIABLES	:	collection(var−dvar)
RELOP	:	atom
LIMIT	:	dvar

Restrictions

required(VALUES, val)
distinct(VALUES, val)
required(VARIABLES, var)
RELOP $\in [=, \neq, <, \geq, >, \leq]$

Purpose Let N be the number of variables of the VARIABLES collection assigned to a value of the VALUES collection. Enforce condition N RELOP LIMIT to hold.

Example

$$\left(\begin{array}{l} \langle 1,3,4,9\rangle, \\ \quad \mathtt{var}-4, \\ \quad \mathtt{var}-5, \\ \left\langle \begin{array}{l} \mathtt{var}-5, \\ \mathtt{var}-4, \\ \mathtt{var}-1, \end{array} \right\rangle, \\ \quad \mathtt{var}-5 \end{array}, =, 3\right)$$

Values 1, 3, 4 and 9 of the VALUES collection are assigned to 3 items of the VARIABLES $= \langle 4,5,5,4,1,5\rangle$ collection. The counts constraint holds since this number is in fact equal (RELOP is set to $=$) to the last argument of the counts constraint.

Typical

|VALUES| > 1
|VARIABLES| > 1
range(VARIABLES.var) > 1
|VARIABLES| > |VALUES|
LIMIT > 0
LIMIT < |VARIABLES|

Symmetries

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

Usage Used in the **Constraint(s) on sets** slot for defining some constraints like assign_and_counts.

Reformulation The count(VALUES, VARIABLES, RELOP , LIMIT) constraint can be expressed in term of the conjunction among(N, VARIABLES, VALUES) $\wedge$ N RELOP LIMIT.

Used in assign_and_counts.

See also **assignment dimension added:** assign_and_counts *(assignment dimension introduced)*.

common keyword: among *(value constraint,counting constraint)*.

specialisation: count *(*variable $\in$ VALUES *replaced by* variable=VALUE*)*.

Keywords **characteristic of a constraint:** automaton, automaton with counters.

constraint network structure: alpha-acyclic constraint network(2).

constraint type: value constraint, counting constraint.

filtering: arc-consistency.

final graph structure: acyclic, bipartite, no loop.

Arc input(s)	VARIABLES VALUES
Arc generator	$PRODUCT \mapsto$ collection(variables, values)
Arc arity	2
Arc constraint(s)	variables.var = values.val
Graph property(ies)	NARC RELOP LIMIT
Graph class	• ACYCLIC • BIPARTITE • NO_LOOP

Graph model Because of the arc constraint variables.var = values.val and since each domain variable can take at most one value, NARC is the number of variables taking a value in the VALUES collection.

Parts (A) and (B) of Figure 5.158 respectively show the initial and final graph associated with the **Example** slot. Since we use the NARC graph property, the arcs of the final graph are stressed in bold.

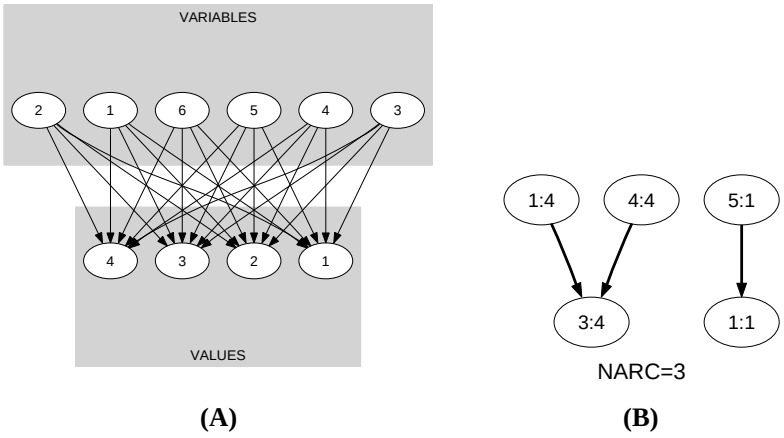

Figure 5.158: Initial and final graph of the counts constraint

Automaton

Figure 5.159 depicts the automaton associated with the `counts` constraint. To each variable $\texttt{VAR}_i$ of the collection `VARIABLES` corresponds a 0-1 signature variable $\texttt{S}_i$. The following signature constraint links $\texttt{VAR}_i$ and $\texttt{S}_i$: $\texttt{VAR}_i \in \texttt{VALUES} \Leftrightarrow \texttt{S}_i$.

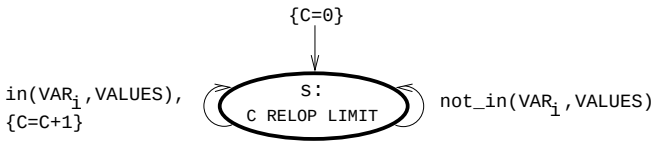

Figure 5.159: Automaton of the `counts` constraint

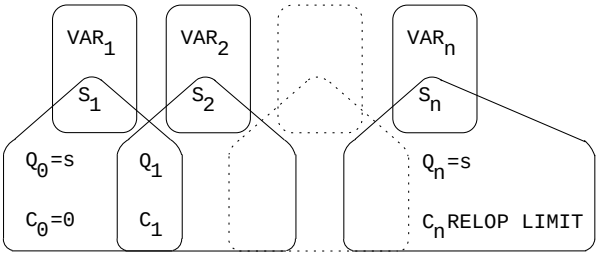

Figure 5.160: Hypergraph of the reformulation corresponding to the automaton of the `counts` constraint