5.91 count

DESCRIPTION LINKS GRAPH AUTOMATON

Origin [99]

Constraint count(VALUE, VARIABLES, RELOP, LIMIT)

Synonyms occurencemax, occurencemin, occurrence.

Arguments

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

Restrictions

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

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

Example $(5, \langle 4, 5, 5, 4, 5\rangle, \geq, 2)$

The count constraint holds since value VALUE $= 5$ occurs 3 times within the items of the collection VARIABLES $= \langle 4, 5, 5, 4, 5\rangle$, which is greater than or equal to (RELOP is set to $\geq$) LIMIT $= 2$.

Typical

|VARIABLES| > 1
range(VARIABLES.var) > 1
RELOP $\in [=, <, \geq, >, \leq]$
LIMIT > 0
LIMIT $<$ |VARIABLES|

Symmetries

- Items of VARIABLES are permutable.
- An occurrence of a value of VARIABLES.var that is different from VALUE can be replaced by any other value that is also different from VALUE.

Arg. properties

- Contractible wrt. VARIABLES when RELOP $\in [<, \leq]$.
- Extensible wrt. VARIABLES when RELOP $\in [\geq, >]$.
- Aggregate: VALUE(id), VARIABLES(union), RELOP(id), LIMIT($+$) when RELOP $\in [<, \leq, \geq, >]$.

Remark Similar to the among constraint. Both, in **JaCoP** (http://www.jacop.eu/) and in **MiniZinc** (http://www.minizinc.org/) RELOP is implicitly set to $=$.

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

Systems occurence in **Choco**, count in **Gecode**, count in **JaCoP**, count in **MiniZinc**, count in **SICStus**.

See also **assignment dimension added:** assign_and_counts *(*variable=VALUE *replaced by* variable $\in$ VALUES *and assignment dimension introduced).*

common keyword: among *(value constraint,counting constraint)*, arith *(value constraint)*, compare_and_count *(counting constraint)*, global_cardinality, max_nvalue, min_nvalue *(value constraint,counting constraint)*, nvalue *(counting constraint)*.

generalisation: counts *(*variable=VALUE *replaced by* variable $\in$ VALUES*).*

related: roots.

used in reformulation: among.

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

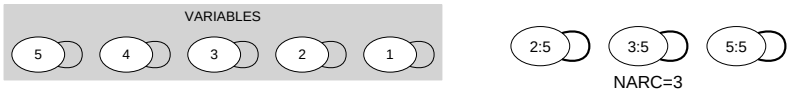
constraint network structure: alpha-acyclic constraint network(2).

constraint type: value constraint, counting constraint.

filtering: arc-consistency.

Arc input(s)	VARIABLES
Arc generator	$SELF \mapsto$ collection(variables)
Arc arity	1
Arc constraint(s)	variables.var = VALUE
Graph property(ies)	**NARC** RELOP LIMIT

Graph model Parts (A) and (B) of Figure 5.220 respectively show the initial and final graph associated with the **Example** slot. Since we use the **NARC** graph property, the loops of the final graph are stressed in bold.



(A) **(B)**

Figure 5.220: Initial and final graph of the count constraint

Automaton

Figure 5.221 depicts the automaton associated with the `count` constraint. To each variable $\texttt{VAR}_i$ of the collection `VARIABLES` corresponds a 0-1 signature variable S_i. The following signature constraint links $\texttt{VAR}_i$ and S_i: $\texttt{VAR}_i = \texttt{VALUE} \Leftrightarrow S_i$.

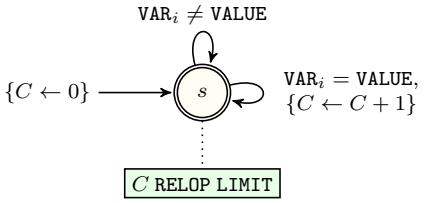


Figure 5.221: Automaton of the `count` constraint

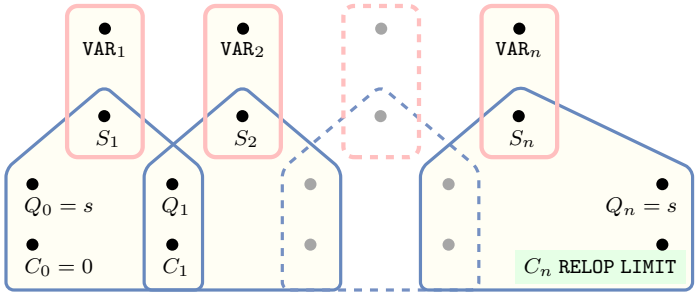


Figure 5.222: Hypergraph of the reformulation corresponding to the automaton (with one counter) of the `count` constraint: since all states variables $Q_0, Q_1, \ldots, Q_n$ are fixed to the unique state s of the automaton, the transitions constraints share only the counter variable C and the constraint network is Berge-acyclic