5.27 among_modulo

DESCRIPTION LINKS GRAPH AUTOMATON

Origin Derived from among.

Constraint among_modulo(NVAR, VARIABLES, REMAINDER, QUOTIENT)

Arguments

NVAR	:	dvar
VARIABLES	:	collection(var−dvar)
REMAINDER	:	int
QUOTIENT	:	int

Restrictions

NVAR ≥ 0
NVAR $\leq$ |VARIABLES|
required(VARIABLES, var)
REMAINDER ≥ 0
REMAINDER $<$ QUOTIENT
QUOTIENT > 0

Purpose NVAR is the number of variables of the collection VARIABLES taking a value that is congruent to REMAINDER modulo QUOTIENT.

Example $(3, \langle 4, 5, 8, 4, 1\rangle, 0, 2)$

In this example REMAINDER $= 0$ and QUOTIENT $= 2$ specifies that we count the number of even values taken by the different variables. As a consequence the among_modulo constraint holds since exactly 3 values of the collection $\langle 4, 5, 8, 4, 1\rangle$ are even.

Typical

NVAR > 0
NVAR $<$ |VARIABLES|
|VARIABLES| > 1
QUOTIENT > 1
QUOTIENT $<$maxval(VARIABLES.var)

Symmetries

- Items of VARIABLES are permutable.
- An occurrence of a value u of VARIABLES.var such that u mod QUOTIENT $=$ REMAINDER (resp. u mod QUOTIENT $\neq$ REMAINDER) can be replaced by any other value v such that v mod QUOTIENT $=$ REMAINDER (resp. u mod QUOTIENT $\neq$ REMAINDER).

Arg. properties

- Functional dependency: NVAR determined by VARIABLES, REMAINDER and QUOTIENT.
- Contractible wrt. VARIABLES when NVAR $= 0$.
- Contractible wrt. VARIABLES when NVAR $=$ |VARIABLES|.
- Aggregate: NVAR(+), VARIABLES(union), REMAINDER(id), QUOTIENT(id).

Remark By giving explicitly all values v that satisfy the equality $v \bmod$ `QUOTIENT` $=$ `REMAINDER`, the `among_modulo` constraint can be modelled with the among constraint. However the `among_modulo` constraint provides a more compact form.

See also **generalisation:** among *(list of* `values` `v` *such that* v$\bmod$`QUOTIENT` = `REMAINDER` *replaced by list of* `values`*)*.

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

constraint arguments: pure functional dependency.

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

constraint type: value constraint, counting constraint.

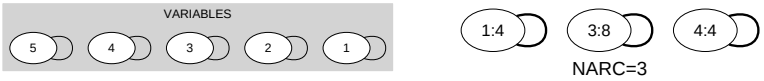
filtering: arc-consistency.

modelling: functional dependency.

Arc input(s)	VARIABLES
Arc generator	$SELF \mapsto$ collection(variables)
Arc arity	1
Arc constraint(s)	variables.var mod QUOTIENT = REMAINDER
Graph property(ies)	NARC= NVAR

Graph model The arc constraint corresponds to a unary constraint. For this reason we employ the *SELF* arc generator in order to produce a graph with a single loop on each vertex.

Parts (A) and (B) of Figure 5.64 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 5.64: Initial and final graph of the among_modulo constraint

Automaton

Figure 5.65 depicts the automaton associated with the among_modulo 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: $\texttt{VAR}_i \bmod \texttt{QUOTIENT} = \texttt{REMAINDER} \Leftrightarrow S_i$.

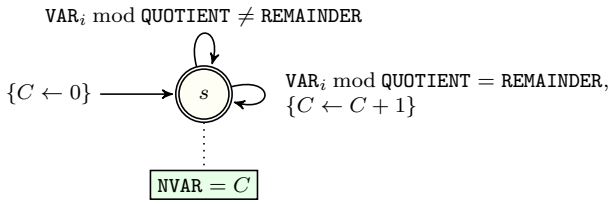


Figure 5.65: Automaton of the among_modulo constraint

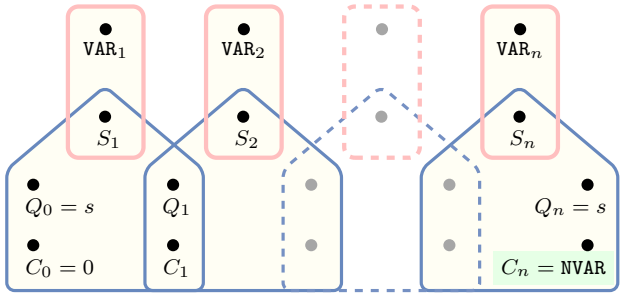


Figure 5.66: Hypergraph of the reformulation corresponding to the automaton (with one counter) of the among_modulo constraint: since all states variables $Q_0, Q_1, \ldots, Q_n$ are fixed to the unique state s of the automaton, the transitions constraints share only the counter variable C and the constraint network is Berge-acyclic