5.25 among_interval

DESCRIPTION LINKS GRAPH AUTOMATON

Origin

Derived from among.

Constraint

among_interval(NVAR, VARIABLES, LOW, UP)

Arguments

NVAR	:	dvar
VARIABLES	:	collection(var−dvar)
LOW	:	int
UP	:	int

Restrictions

$\texttt{NVAR} \geq 0$
$\texttt{NVAR} \leq |\texttt{VARIABLES}|$
required(VARIABLES, var)
$\texttt{LOW} \leq \texttt{UP}$

Purpose

NVAR is the number of variables of the collection VARIABLES taking a value that is located within interval [LOW, UP].

Example

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

The among_interval constraint holds since we have 3 values, namely $4, 5$ and 4 that are situated within interval $[3, 5]$.

All solutions

Figure 5.64 gives all solutions to the following non ground instance of the among_interval constraint: $\texttt{V}_1 \in [2, 9]$, $\texttt{V}_2 \in [0, 1]$, $\texttt{V}_3 \in [5, 6]$, $\texttt{V}_4 \in [1, 2]$, among_interval$(\mathbf{3}, \langle \texttt{V}_1, \texttt{V}_2, \texttt{V}_3, \texttt{V}_4 \rangle, \mathbf{0}, \mathbf{2})$.

① $(\mathbf{3}, \langle \mathbf{2}, \mathbf{0}, 5, \mathbf{1} \rangle, \mathbf{0}, \mathbf{2})$
② $(\mathbf{3}, \langle \mathbf{2}, \mathbf{0}, 5, \mathbf{2} \rangle, \mathbf{0}, \mathbf{2})$
③ $(\mathbf{3}, \langle \mathbf{2}, \mathbf{0}, 6, \mathbf{1} \rangle, \mathbf{0}, \mathbf{2})$
④ $(\mathbf{3}, \langle \mathbf{2}, \mathbf{0}, 6, \mathbf{2} \rangle, \mathbf{0}, \mathbf{2})$
⑤ $(\mathbf{3}, \langle \mathbf{2}, \mathbf{1}, 5, \mathbf{1} \rangle, \mathbf{0}, \mathbf{2})$
⑥ $(\mathbf{3}, \langle \mathbf{2}, \mathbf{1}, 5, \mathbf{2} \rangle, \mathbf{0}, \mathbf{2})$
⑦ $(\mathbf{3}, \langle \mathbf{2}, \mathbf{1}, 6, \mathbf{1} \rangle, \mathbf{0}, \mathbf{2})$
⑧ $(\mathbf{3}, \langle \mathbf{2}, \mathbf{1}, 6, \mathbf{2} \rangle, \mathbf{0}, \mathbf{2})$

Figure 5.64: All solutions corresponding to the non ground example of the among_interval constraint of the **All solutions** slot, where the number of variables assigned a value in $[\texttt{LOW} = \mathbf{0}, \texttt{UP} = \mathbf{2}]$ is equal to $\texttt{NVAR} = \mathbf{3}$

Typical

$\mathtt{NVAR} > 0$
$\mathtt{NVAR} < |\mathtt{VARIABLES}|$
$|\mathtt{VARIABLES}| > 1$
$\mathtt{LOW} < \mathtt{UP}$
$\mathtt{LOW} \leq$ maxval$(\mathtt{VARIABLES.var})$
$\mathtt{UP} \geq$ minval$(\mathtt{VARIABLES.var})$

Symmetries

- Items of VARIABLES are permutable.
- An occurrence of a value of VARIABLES.var that belongs to $[\mathtt{LOW}, \mathtt{UP}]$ (resp. does not belong to $[\mathtt{LOW}, \mathtt{UP}]$) can be replaced by any other value in $[\mathtt{LOW}, \mathtt{UP}]$ (resp. not in $[\mathtt{LOW}, \mathtt{UP}]$).

Arg. properties

- Functional dependency: NVAR determined by VARIABLES, LOW and UP.
- Contractible wrt. VARIABLES when $\mathtt{NVAR} = 0$.
- Contractible wrt. VARIABLES when $\mathtt{NVAR} = |\mathtt{VARIABLES}|$.
- Aggregate: NVAR$(+)$, VARIABLES(union), LOW(id), UP(id).

Remark

By giving explicitly all values of the interval $[\mathtt{LOW}, \mathtt{UP}]$ the among_interval constraint can be modelled with the among constraint. However when $\mathtt{LOW} - \mathtt{UP} + 1$ is a large quantity the among_interval constraint provides a more compact form.

See also

generalisation: among *(*variable *in* interval *replaced by* variable $\in$ values*)*.

Keywords

characteristic of a constraint: 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: interval, functional dependency.

Arc input(s)	`VARIABLES`
Arc generator	*SELF* $\mapsto$ `collection`(`variables`)
Arc arity	1
Arc constraint(s)	• `LOW` $\leq$ `variables.var` • `variables.var` $\leq$ `UP`
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 5.65 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.

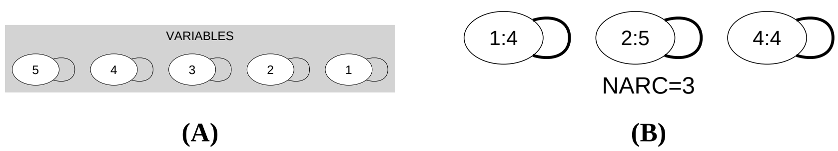

Figure 5.65: Initial and final graph of the `among_interval` constraint

Automaton Figure 5.66 depicts the automaton associated with the `among_interval` constraint. To each variable $\mathtt{VAR}_i$ of the collection `VARIABLES` corresponds a 0-1 signature variable S_i. The following signature constraint links $\mathtt{VAR}_i$ and S_i: $\mathtt{LOW} \leq \mathtt{VAR}_i \wedge \mathtt{VAR}_i \leq \mathtt{UP} \Leftrightarrow S_i$. The automaton counts the number of variables of the `VARIABLES` collection that take their value in $[\mathtt{LOW}, \mathtt{UP}]$ and finally assigns this number to `NVAR`.

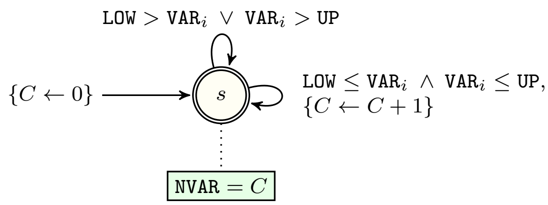

Figure 5.66: Automaton of the `among_interval` constraint

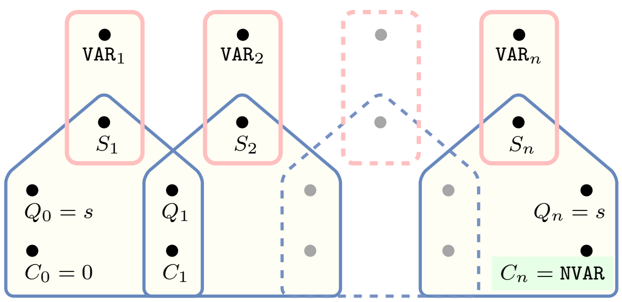

Figure 5.67: Hypergraph of the reformulation corresponding to the automaton (with one counter) of the `among_interval` constraint: since all states variables $Q_0, Q_1, \ldots, Q_n$ are fixed to the unique state s of the automaton, the transitions constraints share only the counter variable C and the constraint network is Berge-acyclic