5.23 among

Origin [41]

Constraint among(NVAR, VARIABLES, VALUES)

Synonyms between, count.

Arguments

```
NVAR      : dvar
VARIABLES : collection(var−dvar)
VALUES    : collection(val−int)
```

Restrictions

NVAR ≥ 0
NVAR $\leq$ |VARIABLES|
required(VARIABLES, var)
required(VALUES, val)
distinct(VALUES, val)

Purpose NVAR is the number of variables of the collection VARIABLES that take their value in VALUES.

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

The among constraint holds since exactly 3 values of the collection of variables $\langle 4, 5, 5, 4, 1\rangle$ belong to the set of values $\{1, 5, 8\}$.

All solutions Figure 5.49 gives all solutions to the following non ground instance of the among constraint: $V_1 \in [1, 5]$, $V_2 \in [3, 9]$, $V_3 \in [5, 6]$, $V_4 \in [2, 3]$, among$(3, \langle V_1, V_2, V_3, V_4\rangle, \langle 2, 4\rangle)$.

① $(3, \langle 2, 4, 5, 2\rangle, \langle 2, 4\rangle)$
② $(3, \langle 2, 4, 6, 2\rangle, \langle 2, 4\rangle)$
③ $(3, \langle 4, 4, 5, 2\rangle, \langle 2, 4\rangle)$
④ $(3, \langle 4, 4, 6, 2\rangle, \langle 2, 4\rangle)$

Figure 5.49: All solutions corresponding to the non ground example of the among constraint of the **All solutions** slot

Typical

NVAR > 0
NVAR $<$ |VARIABLES|
|VARIABLES| > 1
|VALUES| > 1
|VARIABLES| $>$ |VALUES|

Symmetries

- Items of VARIABLES are permutable.
- Items of VALUES 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).

Arg. properties

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

Remark

A similar constraint called between was introduced in **CHIP** in 1990.

The common constraint can be seen as a generalisation of the among constraint where we allow the val attributes of the VALUES collection to be domain variables.

A generalisation of this constraint when the values of VALUES are not initially fixed is called among_var.

When the variable NVAR (i.e., the first argument of the among constraint) does not occur in any other constraints of the problem, it may be operationally more efficient to replace the among constraint by an among_low_up constraint where NVAR is replaced by the corresponding interval $[\underline{\mathtt{NVAR}}, \overline{\mathtt{NVAR}}]$. This stands for two reasons:

- First, by using an among_low_up constraint rather than an among constraint, we avoid the filtering algorithm related to NVAR.
- Second, unlike the among constraint where we need to fix all its variables to get entailment, the among_low_up constraint can be entailed before all its variables get fixed. As a result, this potentially avoid unnecessary calls to its filtering algorithm.

It was shown in [107] that achieving bound-consistency for a conjunction of among constraints where all sets of values are arbitrary intervals can be done in polynomial time.

Algorithm

A filtering algorithm achieving arc-consistency was given by Bessière *et al.* in [61, 64].

Systems

among in **Choco**, count in **Gecode**, among in **JaCoP**, among in **MiniZinc**.

See also

common keyword: arith, atleast, atmost *(value constraint)*, count *(counting constraint)*, counts *(value constraint,counting constraint)*, discrepancy, max_nvalue, min_nvalue, nvalue *(counting constraint)*.

generalisation: among_var *(*constant *replaced by* variable*)*.

implies: among_var, cardinality_atmost.

related: roots *(can be used for expressing* among*)*, sliding_card_skip0 *(counting constraint on maximal sequences)*.

shift of concept: among_seq *(*variable *replaced by* interval *and constraint applied in a sliding way)*, common.

soft variant: open_among *(open constraint)*.

specialisation: `among_diff_0` *(*`variable` $\in$ `values` *replaced by* `variable` *different from* 0*)*, `among_interval` *(*`variable` $\in$ `values` *replaced by* `variable` $\in$ `interval`*)*, `among_low_up` *(*`variable` *replaced by* `interval`*)*, `among_modulo` *(list of* `values` *replaced by list of* `values` `v` *such that* `v` mod `QUOTIENT` = `REMAINDER`*)*, `exactly` *(*`variable` *replaced by* `constant` *and* `values` *replaced by one single* `value`*)*.

system of constraints: `global_cardinality` *(count the number of occurrences of different values)*.

used in graph description: `in`.

uses in its reformulation: `count`.

Keywords

characteristic of a constraint: automaton, automaton with counters, non-deterministic automaton.

constraint arguments: reverse of a constraint, pure functional dependency.

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

constraint type: value constraint, counting constraint.

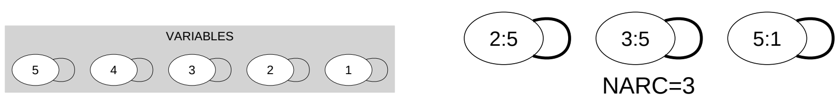
filtering: glue matrix, arc-consistency, SAT.

modelling: functional dependency.

Arc input(s)	VARIABLES
Arc generator	$SELF \mapsto$ collection(variables)
Arc arity	1
Arc constraint(s)	in(variables.var, VALUES)
Graph property(ies)	**NARC** = NVAR

Graph model The arc constraint corresponds to the unary constraint in(variables.var, VALUES) defined in this catalogue. Since this is a unary constraint we employ the *SELF* arc generator in order to produce an initial graph with a single loop on each vertex.

Parts (A) and (B) of Figure 5.50 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 5.50: Initial and final graph of the among constraint

Automaton Figure 5.51 depicts a first automaton that only accepts all the solutions to the among constraint. This automaton uses a counter in order to record the number of satisfied constraints of the form $\mathtt{VAR}_i \in \mathtt{VALUES}$ already encountered. 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: $\mathtt{VAR}_i \in \mathtt{VALUES} \Leftrightarrow S_i$. The automaton counts the number of variables of the VARIABLES collection that take their value in VALUES and finally assigns this number to NVAR.

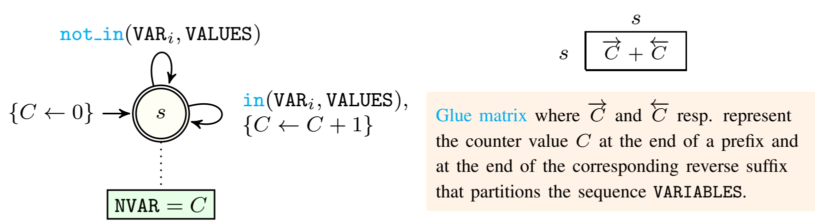


Figure 5.51: Automaton (with one counter) of the among constraint and its glue matrix

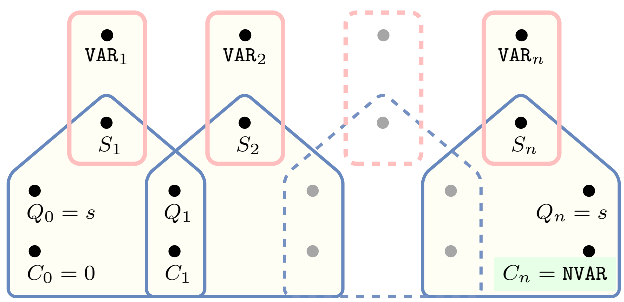


Figure 5.52: Hypergraph of the reformulation corresponding to the automaton (with one counter) of the among constraint: since all states variables $Q_0, Q_1, \ldots, Q_n$ are fixed to the unique state s of the automaton, the transitions constraints share only the counter variable C and the constraint network is Berge-acyclic

We now describe a second counter free automaton that also only accepts all the solutions to among constraint. Without loss of generality, assume that the collection of variables VARIABLES contains at least one variable (i.e., $|\mathtt{VARIABLES}| \geq 1$). Let n and $\mathcal{D}$ respectively denote the number of variables of the collection VARIABLES, and the union of the domains of the variables of VARIABLES. Clearly, the maximum number of variables of VARIABLES that are assigned a value in VALUES cannot exceed the quantity $m = \min(n, \overline{\mathtt{NVAR}})$. The $m+2$ states of the automaton that only accepts all the solutions to the among constraint can be defined in the following way:

- We have an initial state labelled by s_0.
- We have m intermediate states labelled by s_i $(1 \leq i \leq m)$. The intermediate states are indexed by the number of already encountered satisfied constraints of the form $\mathtt{VAR}_k \in \mathtt{VALUES}$ from the initial state s_0 to the state s_i.
- We have an accepting state labelled by s_F.

Three classes of transitions are respectively defined in the following way:

1. There is a transition, labeled by j, $(j \in \mathcal{D} \setminus \texttt{VALUES})$, from every state s_i, $(i \in [0, m])$, to itself.
2. There is a transition, labeled by j, $(j \in \texttt{VALUES})$, from every state s_i, $(i \in [0, m-1])$, to the state s_{i+1}.
3. There is a transition, labelled by i, from every state s_i, $(i \in [0, m])$, to the accepting state s_F.

This leads to an automaton that has $m \cdot |\mathcal{D}| + |\mathcal{D} \setminus \texttt{VALUES}| + m + 1$ transitions. Since the maximum value of m is equal to n, in the worst case we have $n \cdot |\mathcal{D}| + |\mathcal{D} \setminus \texttt{VALUES}| + n + 1$ transitions.

Figure 5.53 depicts a counter free non deterministic automaton associated with the `among` constraint under the hypothesis that (1) all variables of `VARIABLES` are assigned a value in $\{0, 1, 2, 3\}$, (2) $|\texttt{VARIABLES}|$ is equal to 3, (3) `VALUES` corresponds to odd values. The sequence $\texttt{VAR}_1, \texttt{VAR}_2, \ldots, \texttt{VAR}_{|\texttt{VARIABLES}|}, \texttt{NVAR}$ is passed to this automaton. A state s_i $(1 \leq i \leq 3)$ represents the fact that i odd values were already encountered, while s_F represents the accepting state. A transition from s_i $(1 \leq i \leq 3)$ to s_F is labelled by i and represents the fact that we can only go in the accepting state from a state that is compatible with the total number of odd values enforced by `NVAR`. Note that non determinism only occurs if there is a non-empty intersection between the set of potential values that can be assigned to the variables of `VARIABLES` and the potential value of the `NVAR`. While the counter free non deterministic automaton depicted by Figure 5.53 has 5 states and 18 transitions, its minimum-state deterministic counterpart shown in Figure 5.54 has 7 states and 23 transitions.

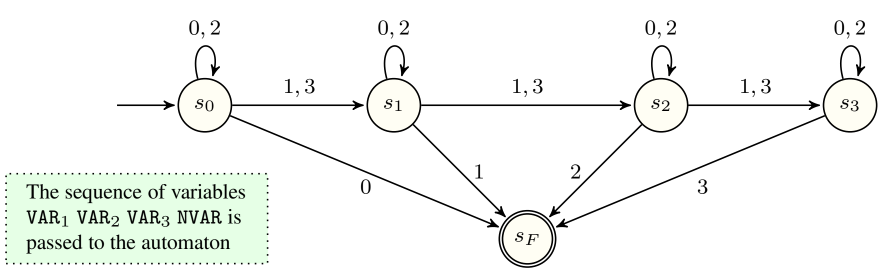


Figure 5.53: Counter free non deterministic automaton of the $\texttt{among}(\texttt{NVAR}, \langle \texttt{VAR}_1, \texttt{VAR}_2, \texttt{VAR}_3 \rangle, \langle 1, 3 \rangle)$ constraint assuming $\texttt{VAR}_i \in [0, 3]$ $(1 \leq i \leq 3)$, with initial state s_0 and accepting state s_F

We make the following final observation. Since the **Symmetries** slot of the `among` constraint indicates that the variables of `VARIABLES` are permutable, and since all incoming transitions to any state of the automaton depicted by Figure 5.53 are labelled with distinct values, we can mechanically construct from this automaton a counter free deterministic automaton that takes as input the sequence `NVAR`, $\texttt{VAR}_3$, $\texttt{VAR}_2$, $\texttt{VAR}_1$ rather than the sequence $\texttt{VAR}_1$, $\texttt{VAR}_2$, $\texttt{VAR}_3$, `NVAR`. This is achieved by respectively making s_F and s_0 the initial and the accepting state, and by reversing each transition.

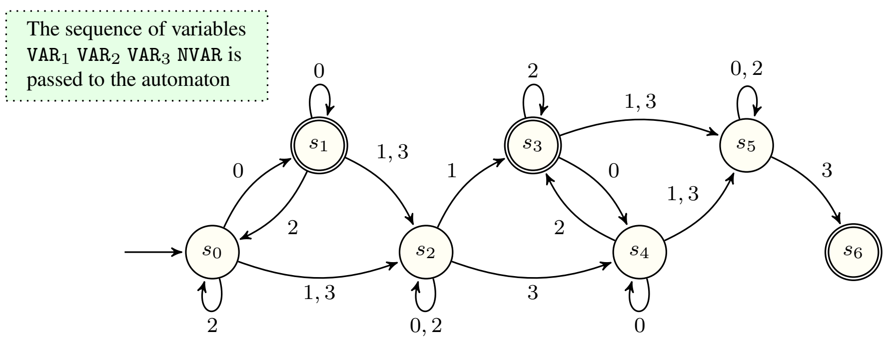


Figure 5.54: Counter free minimum-state deterministic automaton of the $\texttt{among}(\texttt{NVAR}, \langle\texttt{VAR}_1, \texttt{VAR}_2, \texttt{VAR}_3\rangle, \langle 1, 3\rangle)$ constraint assuming $\texttt{VAR}_i \in [0, 3]$ $(1 \leq i \leq 3)$, with initial state s_0 and accepting states s_1, s_3, s_6