

5.26 among_low_up

	DESCRIPTION	LINKS	GRAPH	AUTOMATON
Origin	[41]			
Constraint	among_low_up(LOW, UP, VARIABLES, VALUES)			
Arguments	LOW : int UP : int VARIABLES : collection(var-dvar) VALUES : collection(val-int)			
Restrictions	$LOW \geq 0$ $LOW \leq VARIABLES $ $UP \geq 0$ $UP \leq VARIABLES $ $UP \geq LOW$ required(VARIABLES, var) required(VALUES, val) distinct(VALUES, val)			
Purpose	Between LOW and UP variables of the VARIABLES collection are assigned a value of the VALUES collection.			
Example	(1, 2, (9, 2, 4, 5), (0, 2, 4, 6, 8))			
	The among_low_up constraint holds since between 1 and 2 values (i.e., in fact 2 values) of the collection of values (9, 2, 4, 5) belong to the set of values {0, 2, 4, 6, 8}.			
Typical	$LOW < VARIABLES $ $UP > 0$ $LOW < UP$ $ VARIABLES > 1$ $ VALUES > 1$ $ VARIABLES > VALUES $ $LOW > 0 \vee UP < VARIABLES $			
Symmetries	<ul style="list-style-type: none"> • Items of VARIABLES are permutable. • Items of VALUES are permutable. • LOW can be decreased to any value ≥ 0. • UP can be increased to any value $\leq VARIABLES$. • 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

- **Contractible** wrt. VARIABLES when $UP = 0$.
- **Contractible** wrt. VARIABLES when $UP = |VARIABLES|$.
- **Aggregate**: $LOW(+)$, $UP(+)$, $VARIABLES(\text{union})$, $VALUES(\text{sunion})$.

Algorithm

The `among_low_up` constraint is **entailed** if and only if the following two conditions hold:

1. The number of variables of the VARIABLES collection assigned a value of the VALUES collection is greater than or equal to LOW.
2. The number of variables of the VARIABLES collection that can potentially be assigned a value of the VALUES collection is less than or equal to UP.

Used in

`among_seq`, `cycle_card_on_path`, `interval_and_count`, `sliding_card_skip0`.

See also

assignment dimension added: `interval_and_count` (*assignment dimension corresponding to intervals added*).

generalisation: `among` (*interval replaced by variable*), `sliding_card_skip0` (*full sequence replaced by maximal sequences of non-zeros*).

system of constraints: `among_seq`.

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, entailment.

final graph structure: acyclic, bipartite, no loop.

Cond. implications

`among_low_up`(LOW, UP, VARIABLES, VALUES)
with `distinct`(VARIABLES, var)
implies `among_low_up`(LOW, UP, VALUES, VARIABLES).

Arc input(s)	VARIABLES VALUES
Arc generator	<i>PRODUCT</i> \mapsto collection(variables, values)
Arc arity	2
Arc constraint(s)	variables.var = values.val
Graph property(ies)	<ul style="list-style-type: none"> • NARC \geq LOW • NARC \leq UP
Graph class	<ul style="list-style-type: none"> • ACYCLIC • BIPARTITE • NO_LOOP

Graph model

Each arc constraint of the final graph corresponds to the fact that a variable is assigned to a value that belong to the VALUES collection. The two graph properties restrict the total number of arcs to the interval [LOW, UP].

Parts (A) and (B) of Figure 5.61 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.

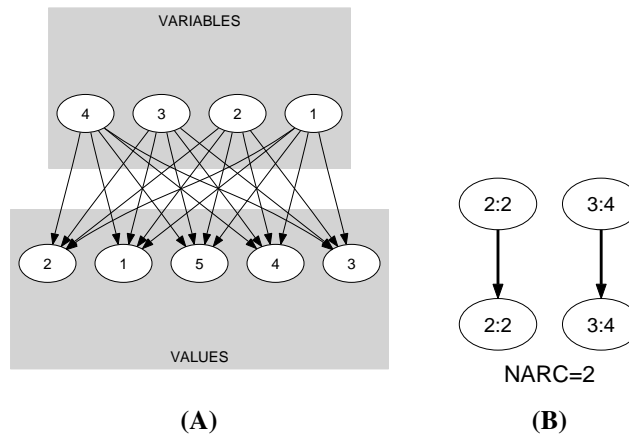


Figure 5.61: Initial and final graph of the among_low_up constraint

Automaton

Figure 5.62 depicts the automaton associated with the `among_low_up` constraint. To each variable VAR_i of the collection `VARIABLES` corresponds a 0-1 signature variable S_i . The following signature constraint links VAR_i and S_i : $VAR_i \in VALUES \Leftrightarrow S_i$. The automaton counts the number of variables of the `VARIABLES` collection that take their value in `VALUES` and finally checks that this number is within the interval $[LOW, UP]$.

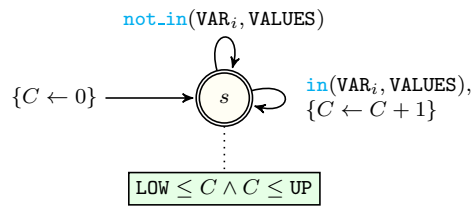


Figure 5.62: Automaton of the `among_low_up` constraint

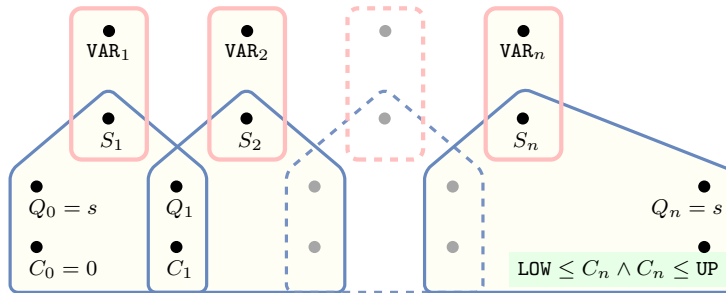


Figure 5.63: Hypergraph of the reformulation corresponding to the automaton (with one counter) of the `among_low_up` constraint: since all states variables Q_0, Q_1, \dots, Q_n are fixed to the unique state s of the automaton, the transitions constraints share only the counter variable C and the constraint network is Berge-acyclic