

## 5.77 count

	DESCRIPTION	LINKS	GRAPH	AUTOMATON
<b>Origin</b>	[90]			
<b>Constraint</b>	count(VALUE, VARIABLES, RELOP, LIMIT)			
<b>Synonyms</b>	occurencemax, occurencemin, occurrence.			
<b>Arguments</b>	VALUE : int VARIABLES : collection(var-dvar) RELOP : atom LIMIT : dvar			
<b>Restrictions</b>	required(VARIABLES, var) RELOP ∈ [=, ≠, <, ≥, >, ≤]			
<b>Purpose</b>	Let $N$ be the number of variables of the VARIABLES collection assigned to value VALUE; Enforce condition $N$ RELOP LIMIT to hold.			
<b>Example</b>	$( 5, \langle 4, 5, 5, 4, 5 \rangle, \geq, 2 )$			
	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.			
<b>Typical</b>	VARIABLES  > 1 range(VARIABLES.var) > 1 LIMIT > 0 LIMIT <  VARIABLES			
<b>Symmetries</b>	<ul style="list-style-type: none"> <li>Items of VARIABLES are <a href="#">permutable</a>.</li> <li>An occurrence of a value of VARIABLES.var that is different from VALUE can be <a href="#">replaced</a> by any other value that is also different from VALUE.</li> </ul>			
<b>Remark</b>	Similar to the <a href="#">among</a> constraint. In <b>JaCoP</b> ( <a href="http://www.jacop.eu/">http://www.jacop.eu/</a> ) RELOP is implicitly set to =.			
<b>Reformulation</b>	The count(VALUE, VARIABLES, RELOP, LIMIT) constraint can be expressed in term of the conjunction <a href="#">among</a> ( $N$ , VARIABLES, (VALUE)) $\wedge$ $N$ RELOP LIMIT.			
<b>Systems</b>	occurence in <b>Choco</b> , count in <b>Gecode</b> , count in <b>JaCoP</b> , count in <b>SICStus</b> .			

**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), `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.

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

**Graph model**

Parts (A) and (B) of Figure 5.155 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.

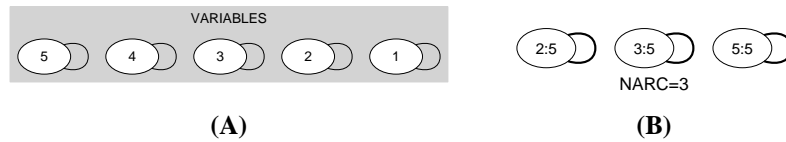


Figure 5.155: Initial and final graph of the count constraint

**Automaton**

Figure 5.156 depicts the automaton associated with the count 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  $S_i$ :  $VAR_i = VALUE \Leftrightarrow S_i$ .

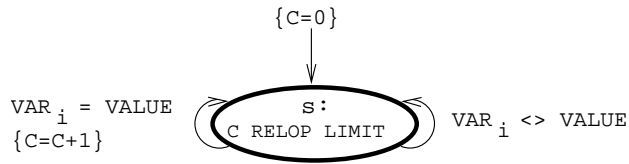


Figure 5.156: Automaton of the count constraint

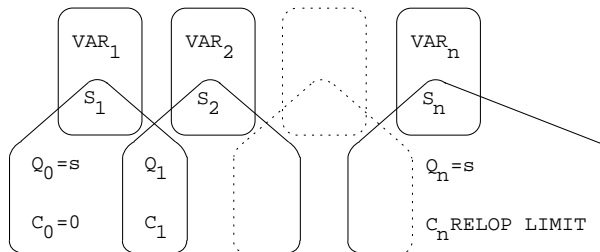


Figure 5.157: Hypergraph of the reformulation corresponding to the automaton of the count constraint