

### 5.35 `atmost_nvector`

	DESCRIPTION	LINKS	GRAPH
<b>Origin</b>	Derived from <code>nvector</code>		
<b>Constraint</b>	<code>atmost_nvector(NVEC, VECTORS)</code>		
<b>Type</b>	<code>VECTOR</code> : <code>collection(var-dvar)</code>		
<b>Arguments</b>	<code>NVEC</code> : <code>dvar</code> <code>VECTORS</code> : <code>collection(vec - VECTOR)</code>		
<b>Restrictions</b>	<code>NVEC</code> $\geq$ <code>min(1,  VECTORS )</code> <code>required(VECTORS, vec)</code> <code>same_size(VECTORS, vec)</code>		
<b>Purpose</b>	The number of distinct tuples of values taken by the vectors of the collection <code>VECTORS</code> is less than or equal to <code>NVEC</code> . Two tuples of values $\langle A_1, A_2, \dots, A_m \rangle$ and $\langle B_1, B_2, \dots, B_m \rangle$ are <i>distinct</i> if and only if there exist an integer $i \in [1, m]$ such that $A_i \neq B_i$ .		
<b>Example</b>	$\left( 3, \left\langle \begin{array}{l} \text{vec} - \langle 5, 6 \rangle, \\ \text{vec} - \langle 5, 6 \rangle, \\ \text{vec} - \langle 9, 3 \rangle, \\ \text{vec} - \langle 5, 6 \rangle, \\ \text{vec} - \langle 9, 3 \rangle \end{array} \right\rangle \right)$		
	The <code>atmost_nvector</code> constraint holds since the collection <code>VECTORS</code> involves at most 3 distinct tuples of values (i.e., in fact the 2 distinct tuples $\langle 5, 6 \rangle$ and $\langle 9, 3 \rangle$ ).		
<b>Typical</b>	<code>NVEC</code> $>$ 1 <code>NVEC</code> $<$ <code> VECTORS </code> <code> VECTOR </code> $>$ 1 <code> VECTORS </code> $>$ 1		
<b>Symmetries</b>	<ul style="list-style-type: none"> <li>• <code>NVEC</code> can be <b>increased</b>.</li> <li>• Items of <code>VECTORS</code> are <b>permutable</b>.</li> <li>• Items of <code>VECTORS.vec</code> are <b>permutable</b> (<i>same permutation used</i>).</li> <li>• All occurrences of two distinct tuples of values of <code>VECTORS.vec</code> can be <b>swapped</b>; all occurrences of a tuple of values of <code>VECTORS.vec</code> can be <b>renamed</b> to any unused tuple of values.</li> </ul>		
<b>Reformulation</b>	By introducing an extra variable <code>NV</code> $\in [0,  \text{VECTORS} ]$ , the <code>atmost_nvector(NV, VECTORS)</code> constraint can be expressed in term of an <code>nvector(NV, VECTORS)</code> constraint and of an inequality constraint <code>NV</code> $\leq$ <code>NVEC</code> .		

**See also**

**comparison swapped:** `atleast_nvector`.

**generalisation:** `nvector` ( $\leq$  NVEC replaced by  $=$  NVEC).

**implied by:** `ordered_atmost_nvector`.

**used in graph description:** `lex_equal`.

**Keywords**

**characteristic of a constraint:** `vector`.

**constraint type:** counting constraint, value partitioning constraint.

**final graph structure:** strongly connected component, equivalence.

**modelling:** number of distinct equivalence classes.

**problems:** domination.

<b>Arc input(s)</b>	VECTORS
<b>Arc generator</b>	<i>CLIQUE</i> $\mapsto$ <code>collection</code> (vectors1, vectors2)
<b>Arc arity</b>	2
<b>Arc constraint(s)</b>	<code>lex_equal</code> (vectors1.vec, vectors2.vec)
<b>Graph property(ies)</b>	$NSCC \leq NVEC$
<b>Graph class</b>	<u>EQUIVALENCE</u>

**Graph model**

Parts (A) and (B) of Figure 5.64 respectively show the initial and final graph associated with the **Example** slot. Since we use the **NSCC** graph property we show the different strongly connected components of the final graph. Each strongly connected component corresponds to a tuple of values that is assigned to some vectors of the **VECTORS** collection. The 2 following tuple of values  $\langle 5, 6 \rangle$  and  $\langle 9, 3 \rangle$  are used by the vectors of the **VECTORS** collection.

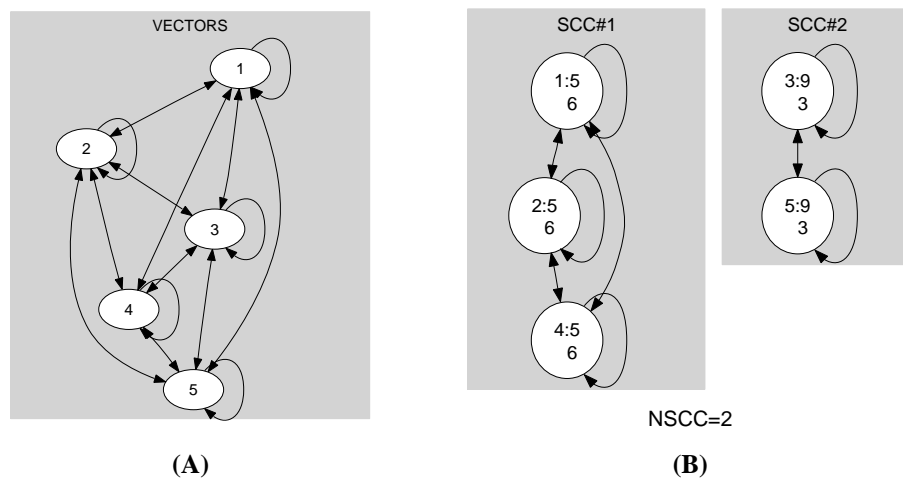


Figure 5.64: Initial and final graph of the `atleast_nvector` constraint

