## 5.38 atleast_nvector

DESCRIPTION LINKS GRAPH

**Origin** Derived from nvector

**Constraint** atleast_nvector(NVEC, VECTORS)

**Type**
```
VECTOR : collection(var−dvar)
```

**Arguments**
```
NVEC    : dvar
VECTORS : collection(vec − VECTOR)
```

**Restrictions**
```
|VECTOR| ≥ 1
NVEC ≥ 0
NVEC ≤ |VECTORS|
required(VECTORS, vec)
same_size(VECTORS, vec)
```

**Purpose** The number of distinct tuples of values taken by the vectors of the collection VECTORS is greater than or equal to NVEC. Two tuples of values $\langle A_1, A_2, \ldots, A_m \rangle$ and $\langle B_1, B_2, \ldots, B_m \rangle$ are *distinct* if and only if there exist an integer $i \in [1, m]$ such that $A_i \neq B_i$.

**Example**

$$\left( 2, \left\langle \begin{array}{l} \texttt{vec} - \langle 5, 6 \rangle, \\ \texttt{vec} - \langle 5, 6 \rangle, \\ \texttt{vec} - \langle 9, 3 \rangle, \\ \texttt{vec} - \langle 5, 6 \rangle, \\ \texttt{vec} - \langle 9, 4 \rangle \end{array} \right\rangle \right)$$

The atleast_nvector constraint holds since the collection VECTORS involves at least 2 distinct tuples of values (i.e., in fact the 3 distinct tuples $\langle 5, 6 \rangle$, $\langle 9, 3 \rangle$ and $\langle 9, 4 \rangle$).

**Typical**
```
|VECTOR| > 1
NVEC > 1
NVEC < |VECTORS|
|VECTORS| > 1
```

**Symmetries**
- NVEC can be decreased to any value ≥ 0.
- Items of VECTORS are permutable.
- Items of VECTORS.vec are permutable (*same permutation used*).
- All occurrences of two distinct tuples of values of VECTORS.vec can be swapped; all occurrences of a tuple of values of VECTORS.vec can be renamed to any unused tuple of values.

**Arg. properties** Extensible wrt. VECTORS.

**Reformulation** By introducing an extra variable NV $\in [0, |\texttt{VECTORS}|]$, the atleast_nvector(NV, VECTORS) constraint can be expressed in term of an nvector(NV, VECTORS) constraint and of an inequality constraint NV $\geq$ NVEC.

**See also** **comparison swapped:** atmost_nvector.

**implied by:** nvector *(*$\geq$ NVEC *replaced by* $=$ NVEC*)*, ordered_atleast_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.

| | |
|---|---|
| **Arc input(s)** | VECTORS |
| **Arc generator** | $CLIQUE \mapsto$ collection(vectors1, vectors2) |
| **Arc arity** | 2 |
| **Arc constraint(s)** | lex_equal(vectors1.vec, vectors2.vec) |
| **Graph property(ies)** | NSCC $\geq$ NVEC |
| **Graph class** | EQUIVALENCE |

**Graph model** Parts (A) and (B) of Figure 5.90 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 3 following tuple of values $\langle 5, 6\rangle$, $\langle 9, 3\rangle$ and $\langle 9, 4\rangle$ are used by the vectors of the VECTORS collection.

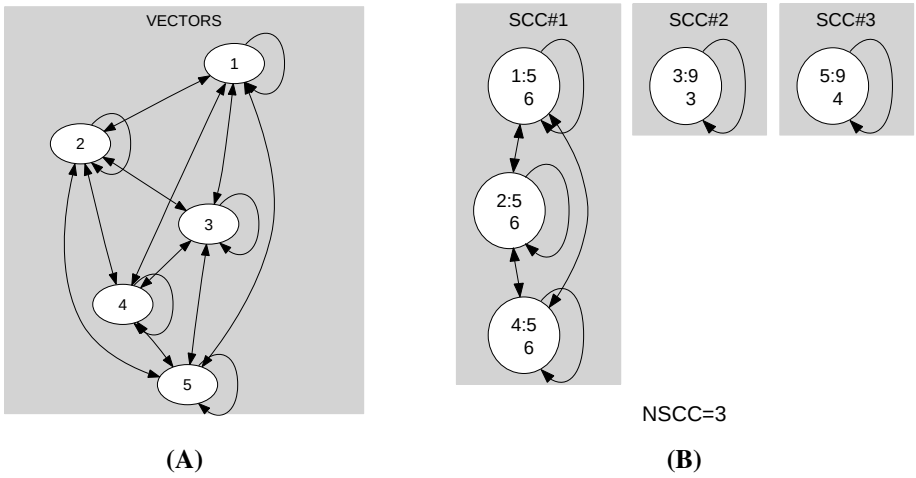


Figure 5.90: Initial and final graph of the atleast_nvector constraint