5.278 ninterval

Origin   Derived from nvalue.

Constraint   ninterval(NVAL, VARIABLES, SIZE_INTERVAL)

Arguments
```
NVAL          :  dvar
VARIABLES     :  collection(var−dvar)
SIZE_INTERVAL :  int
```

Restrictions
NVAL ≥ min(1, |VARIABLES|)
NVAL ≤ |VARIABLES|
required(VARIABLES, var)
SIZE_INTERVAL > 0

Purpose
Consider the intervals of the form $[\text{SIZE\_INTERVAL} \cdot k, \text{SIZE\_INTERVAL} \cdot k + \text{SIZE\_INTERVAL} - 1]$ where k is an integer. NVAL is the number of intervals for which at least one value is assigned to at least one variable of the collection VARIABLES.

Example
$(2, \langle 3, 1, 9, 1, 9 \rangle, 4)$

In the example, the third argument SIZE_INTERVAL $=$ 4 defines the following family of intervals $[4 \cdot k, 4 \cdot k + 3]$, where k is an integer. Values 3, 1, 9, 1 and 9 are respectively located within intervals $[0, 3]$, $[0, 3]$, $[8, 11]$, $[0, 3]$ and $[8, 11]$. Since we only use the two intervals $[0, 3]$ and $[8, 11]$ the first argument of the ninterval constraint is set to value 2.

Typical
NVAL > 1
NVAL < |VARIABLES|
SIZE_INTERVAL > 1
SIZE_INTERVAL < range(VARIABLES.var)
(nval(VARIABLES.var) + SIZE_INTERVAL − 1)/SIZE_INTERVAL < NVAL

Symmetries
- Items of VARIABLES are permutable.
- An occurrence of a value of VARIABLES.var that belongs to the k-th interval, of size SIZE_INTERVAL, can be replaced by any other value of the same interval.

Arg. properties
- Functional dependency: NVAL determined by VARIABLES and SIZE_INTERVAL.
- Contractible wrt. VARIABLES when NVAL = 1 and |VARIABLES| > 0.
- Contractible wrt. VARIABLES when NVAL = |VARIABLES|.

Usage
The ninterval constraint is useful for counting the number of actually used periods, no matter how many time each period is used. A period can for example stand for a hour or for a day.

Algorithm   [27, 40].

See also   **related:** nclass *(variable/constant* replaced *by* variable ∈ partition*)*,
                      nequivalence *(variable/constant* replaced *by* variable mod constant*)*,
                      npair *(variable/constant* replaced by pair *of* variables*)*.

                      specialisation: nvalue *(variable/constant* replaced by variable*)*.

Keywords   **constraint arguments:** pure functional dependency.

                      constraint type: counting constraint, value partitioning constraint.

                      final graph structure: strongly connected component, equivalence.

                      modelling: number of distinct equivalence classes, interval, functional dependency.

Arc input(s)	VARIABLES
Arc generator	*CLIQUE* $\mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	variables1.var/SIZE_INTERVAL = variables2.var/SIZE_INTERVAL
Graph property(ies)	**NSCC**= NVAL

Graph model   Parts (A) and (B) of Figure 5.598 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 those values of an interval that are assigned to some variables of the VARIABLES collection. The values 1, 3 and the value 9, which respectively correspond to intervals $[0, 3]$ and $[8, 11]$, are assigned to the variables of the VARIABLES collection.
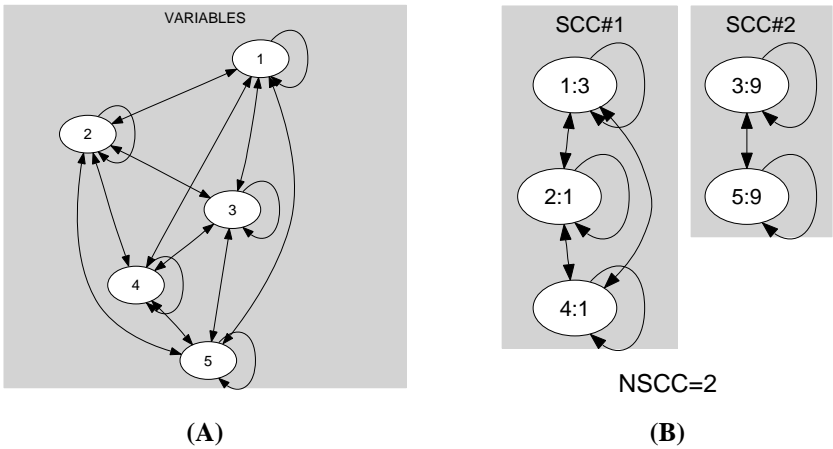

(A)   (B)

Figure 5.598: Initial and final graph of the ninterval constraint