5.90 correspondence

DESCRIPTION LINKS GRAPH

Origin Derived from sort_permutation by removing the sorting condition.

Constraint correspondence(FROM, PERMUTATION, TO)

Arguments

FROM	:	collection(from−dvar)
PERMUTATION	:	collection(var−dvar)
TO	:	collection(tvar−dvar)

Restrictions

|PERMUTATION| = |FROM|
|PERMUTATION| = |TO|
PERMUTATION.var ≥ 1
PERMUTATION.var ≤ |PERMUTATION|
alldifferent(PERMUTATION)
required(FROM, from)
required(PERMUTATION, var)
required(TO, tvar)

Purpose The variables of collection FROM correspond to the variables of collection TO according to the permutation PERMUTATION (i.e., FROM[i].from = TO[PERMUTATION[i].var].tvar).

Example $(\langle 1,9,1,5,2,1\rangle, \langle 6,1,3,5,4,2\rangle, \langle 9,1,1,2,5,1\rangle)$

As illustrated by Figure 5.218, the correspondence constraint holds since:

- The first item FROM[1].from = 1 of collection FROM corresponds to the PERMUTATION[1].var = 6^{th} item of collection TO.
- The second item FROM[2].from = 9 of collection FROM corresponds to the PERMUTATION[2].var = 1^{th} item of collection TO.
- The third item FROM[3].from = 1 of collection FROM corresponds to the PERMUTATION[3].var = 3^{th} item of collection TO.
- The fourth item FROM[4].from = 5 of collection FROM corresponds to the PERMUTATION[4].var = 5^{th} item of collection TO.
- The fifth item FROM[5].from = 2 of collection FROM corresponds to the PERMUTATION[5].var = 4^{th} item of collection TO.
- The sixth item FROM[6].from = 1 of collection FROM corresponds to the PERMUTATION[6].var = 2^{th} item of collection TO.

Typical

|FROM| > 1
range(FROM.from) > 1

Symmetry All occurrences of two distinct values in FROM.from or TO.tvar can be swapped; all occurrences of a value in FROM.from or TO.tvar can be renamed to any unused value.

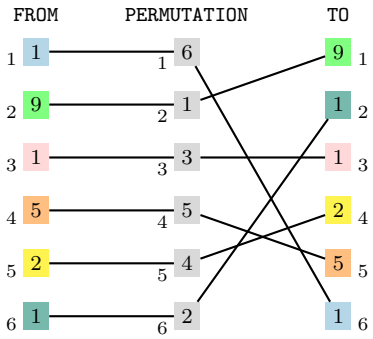


Figure 5.218: Illustration of the correspondence between the items of the FROM and the TO collections according to the permutation defined by the items of the PERMUTATION collection of the **Example** slot

Remark Similar to the same constraint except that we also provide the permutation that allows to go from the items of collection FROM to the items of collection TO.

Algorithm An arc-consistency filtering algorithm for the correspondence constraint is described in [129, 130]. The algorithm is based on the following ideas:

- First, one can map solutions to the correspondence constraint to perfect matchings in a bipartite graph derived from the domain of the variables of the constraint in the following way: to each variable of the FROM collection there is a *from* vertex; similarly, to each variable of the TO collection there is a *to* vertex; finally, there is an edge between the i^{th} from vertex and the j^{th} to vertex if and only if the corresponding domains intersect and if j belongs to the domain of the i^{th} permutation variable.
- Second, Dulmage-Mendelsohn decomposition [148] is used to characterise all edges that do not belong to any perfect matching, and therefore prune the corresponding variables.

See also **implied by:** sort_permutation.

specialisation: same *(*PERMUTATION *parameter removed)*.

Keywords **characteristic of a constraint:** derived collection.

combinatorial object: permutation.

constraint arguments: constraint between three collections of variables.

filtering: bipartite matching.

final graph structure: acyclic, bipartite, no loop.

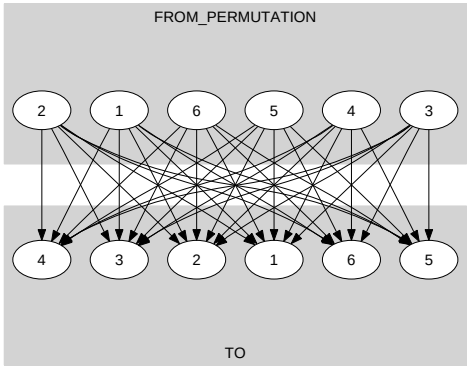
Derived Collection

$$\texttt{col}\left(\begin{array}{l}\texttt{FROM\_PERMUTATION}-\texttt{collection}(\texttt{from}-\texttt{dvar},\texttt{var}-\texttt{dvar}),\\ [\texttt{item}(\texttt{from}-\texttt{FROM.from},\texttt{var}-\texttt{PERMUTATION.var})]\end{array}\right)$$

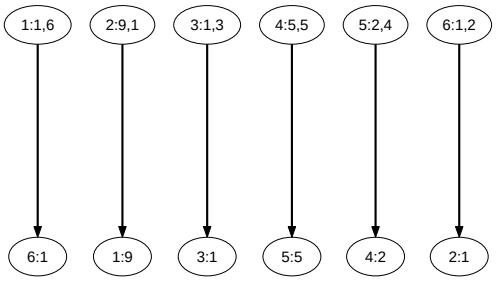
Arc input(s)	FROM_PERMUTATION TO
Arc generator	$PRODUCT \mapsto \texttt{collection}(\texttt{from\_permutation},\texttt{to})$
Arc arity	2
Arc constraint(s)	• from_permutation.from = to.tvar • from_permutation.var = to.key
Graph property(ies)	NARC = |PERMUTATION|
Graph class	• ACYCLIC • BIPARTITE • NO_LOOP

Graph model Parts (A) and (B) of Figure 5.219 respectively show the initial and final graph associated with the **Example** slot. In both graphs the source vertices correspond to the derived collection FROM_PERMUTATION, while the sink vertices correspond to the collection TO. Since the final graph contains exactly |PERMUTATION| arcs the correspondence constraint holds. As we use the NARC graph property, the arcs of the final graph are stressed in bold.

Signature Because of the second condition from_permutation.var = to.key of the arc constraint and since both, the var attributes of the collection FROM_PERMUTATION and the key attributes of the collection TO are all-distinct, the final graph contains at most |PERMUTATION| arcs. Therefore we can rewrite the graph property NARC = |PERMUTATION| to NARC $\geq$ |PERMUTATION|. This leads to simplify $\overline{\text{NARC}}$ to NARC.



(A)



(B)

Figure 5.219: Initial and final graph of the `correspondence` constraint