5.371 sort

DESCRIPTION **LINKS** **GRAPH**

Origin [297]

Constraint `sort(VARIABLES1, VARIABLES2)`

Synonyms `sortedness`, `sorted`, `sorting`.

Arguments

`VARIABLES1 : collection(var−dvar)`
`VARIABLES2 : collection(var−dvar)`

Restrictions

`|VARIABLES1| = |VARIABLES2|`
`required(VARIABLES1, var)`
`required(VARIABLES2, var)`

Purpose First, the variables of the collection `VARIABLES2` correspond to a permutation of the variables of `VARIABLES1`. Second, the variables of `VARIABLES2` are sorted in increasing order.

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

The `sort` constraint holds since:

- Values 1, 2, 5 and 9 have the same number of occurrences within both collections $\langle 1, 9, 1, 5, 2, 1\rangle$ and $\langle 1, 1, 1, 2, 5, 9\rangle$. Figure 5.715 illustrates this correspondence.
- The items of collection $\langle 1, 1, 1, 2, 5, 9\rangle$ are sorted in increasing order.

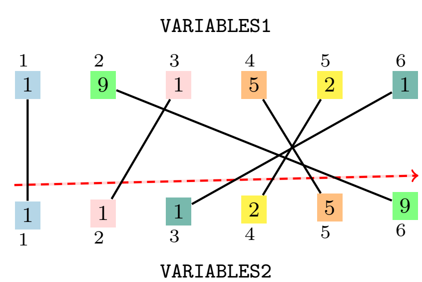


Figure 5.715: Illustration of the correspondence between the items of the `VARIABLES1` and of the `VARIABLES2` collections of the **Example** slot (note that the items of the `VARIABLES2` are sorted in increasing order)

All solutions Figure 5.716 gives all solutions to the following non ground instance of the `sort` constraint: $V_1 \in [2, 3]$, $V_2 \in [2, 3]$, $V_3 \in [1, 2]$, $V_4 \in [4, 5]$, $V_5 \in [2, 4]$, $S_1 \in [2, 3]$, $S_2 \in [2, 3]$, $S_3 \in [1, 3]$, $S_4 \in [4, 5]$, $S_5 \in [2, 5]$, `sort`$(\langle V_1, V_2, V_3, V_4, V_5\rangle, \langle S_1, S_2, S_3, S_4, S_5\rangle)$.

① $(\langle 2,2,2,4,4\rangle, \langle 2,2,2,4,4\rangle)$
② $(\langle 2,2,2,5,4\rangle, \langle 2,2,2,4,5\rangle)$
③ $(\langle 2,3,2,4,4\rangle, \langle 2,2,3,4,4\rangle)$
④ $(\langle 2,3,2,5,4\rangle, \langle 2,2,3,4,5\rangle)$
⑤ $(\langle 3,2,2,4,4\rangle, \langle 2,2,3,4,4\rangle)$
⑥ $(\langle 3,2,2,5,4\rangle, \langle 2,2,3,4,5\rangle)$
⑦ $(\langle 3,3,2,4,4\rangle, \langle 2,3,3,4,4\rangle)$
⑧ $(\langle 3,3,2,5,4\rangle, \langle 2,3,3,4,5\rangle)$

Figure 5.716: All solutions corresponding to the non ground example of the `sort` constraint of the **All solutions** slot

Typical

$|\texttt{VARIABLES1}| > 1$
$\texttt{range}(\texttt{VARIABLES1.var}) > 1$

Symmetries

- Items of `VARIABLES1` are permutable.
- One and the same constant can be added to the `var` attributes of all items of `VARIABLES1` and `VARIABLES2`.

Arg. properties

Functional dependency: `VARIABLES2` determined by `VARIABLES1`.

Usage

The main usage of the `sort` constraint, that was not foreseen when the `sort` constraint was invented, is its use in many reformulations. Many constraints involving one or several collections of variables *become much simpler to express when the variables of these collections are sorted*. In addition these reformulations typically have a size that is linear in the number of variables of the original constraint. This justifies why the `sort` constraint is considered to be a core constraint. As illustrative examples of these types of reformulations we successively consider the `alldifferent` and the `same` constraints:

- The `alldifferent`$(\langle v_1, v_2, \ldots, v_n\rangle)$ constraint can be reformulated as the conjunction `sort`$(\langle v_1, v_2, \ldots, v_n\rangle, \langle w_1, w_2, \ldots, w_n\rangle)$ $\wedge$ `strictly_increasing`$(\langle w_1, w_2, \ldots, w_n\rangle)$.
- The `same`$(\langle u_1, u_2, \ldots, u_n\rangle, \langle v_1, v_2, \ldots, v_n\rangle)$ constraint can be reformulated as the conjunction `sort`$(\langle u_1, u_2, \ldots, u_n\rangle, \langle w_1, w_2, \ldots, w_n\rangle)$ $\wedge$ `sort`$(\langle v_1, v_2, \ldots, v_n\rangle, \langle w_1, w_2, \ldots, w_n\rangle)$.

Remark

A variant of this constraint called `sort_permutation` was introduced in [449]. In this variant an additional list of domain variables represents the permutation that allows to go from `VARIABLES1` to `VARIABLES2`.

Algorithm

[78, 281].

Systems

sorting in **Choco**, sorted in **Gecode**, sort in **MiniZinc**, sorting in **SICStus**.

See also

generalisation: `sort_permutation` *(*`PERMUTATION` *parameter added)*.

implies: `lex_greatereq`, `same`.

uses in its reformulation: `alldifferent`, `same`.

Keywords

characteristic of a constraint: core, sort.

combinatorial object: permutation.

constraint arguments: constraint between two collections of variables, pure functional dependency.

filtering: bound-consistency.

modelling: functional dependency.

Arc input(s)	VARIABLES1 VARIABLES2
Arc generator	$PRODUCT \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	variables1.var = variables2.var
Graph property(ies)	• for all connected components: **NSOURCE**=**NSINK** • **NSOURCE**= |VARIABLES1| • **NSINK**= |VARIABLES2|

Arc input(s)	VARIABLES2
Arc generator	$PATH \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	variables1.var $\leq$ variables2.var
Graph property(ies)	**NARC**= |VARIABLES2| $-$ 1

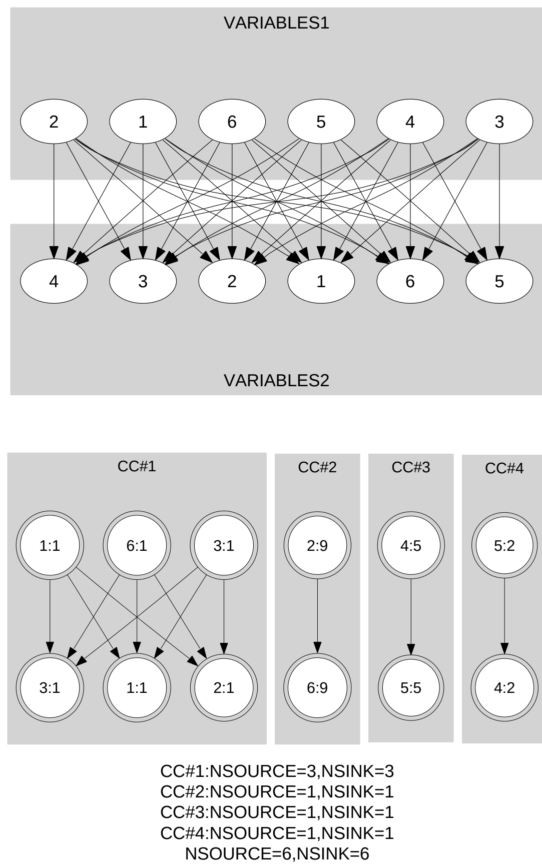
Graph model Parts (A) and (B) of Figure 5.717 respectively show the initial and final graph associated with the first graph constraint of the **Example** slot. Since it uses the **NSOURCE** and **NSINK** graph properties, the source and sink vertices of this final graph are stressed with a double circle. Since there is a constraint on each connected component of the final graph we also show the different connected components. The sort constraint holds since:

- Each connected component of the final graph of the first graph constraint has the same number of sources and of sinks.
- The number of sources of the final graph of the first graph constraint is equal to |VARIABLES1|.
- The number of sinks of the final graph of the first graph constraint is equal to |VARIABLES2|.
- Finally the second graph constraint holds also since its corresponding final graph contains exactly |VARIABLES1 $-$ 1| arcs: all the inequalities constraints between consecutive variables of VARIABLES2 holds.

Signature Consider the first graph constraint. Since the initial graph contains only sources and sinks, and since isolated vertices are eliminated from the final graph, we make the following observations:

- Sources of the initial graph cannot become sinks of the final graph,
- Sinks of the initial graph cannot become sources of the final graph.

From the previous observations and since we use the $PRODUCT$ arc generator on the collections VARIABLES1 and VARIABLES2, we have that the maximum number of sources and sinks of the final graph is respectively equal to |VARIABLES1| and |VARIABLES2|. Therefore we can rewrite **NSOURCE** = |VARIABLES1| to **NSOURCE** $\geq$ |VARIABLES1| and simplify $\underline{\overline{\mathbf{NSOURCE}}}$ to $\overline{\mathbf{NSOURCE}}$. In a similar way, we can rewrite **NSINK** = |VARIABLES2| to **NSINK** $\geq$ |VARIABLES2| and simplify $\underline{\overline{\mathbf{NSINK}}}$ to $\overline{\mathbf{NSINK}}$.



(A)

(B)

Figure 5.717: Initial and final graph of the `sort` constraint

Consider now the second graph constraint. Since we use the $PATH$ arc generator with an arity of 2 on the `VARIABLES2` collection, the maximum number of arcs of the final graph is equal to $|\texttt{VARIABLES2}| - 1$. Therefore we can rewrite the graph property $\mathbf{NARC} = |\texttt{VARIABLES2}| - 1$ to $\mathbf{NARC} \geq |\texttt{VARIABLES2}| - 1$ and simplify $\underline{\overline{\mathbf{NARC}}}$ to $\overline{\mathbf{NARC}}$.

Quiz

EXERCISE 1 (checking whether a ground instance holds or not)[a]

A. Does the constraint $\texttt{sort}(\langle 1, 0, 0, 1\rangle, \langle 0, 0, 1\rangle)$ hold?

B. Does the constraint $\texttt{sort}(\langle 3, 5, 3, 1\rangle, \langle 1, 3, 5\rangle)$ hold?

C. Does the constraint $\texttt{sort}(\langle 2, 4, 2, 2, 4\rangle, \langle 2, 2, 2, 4, 4\rangle)$ hold?

D. Does the constraint $\texttt{sort}(\langle 2, 4, 2, 2, 4\rangle, \langle 4, 4, 2, 2, 2\rangle)$ hold?

[a]Hint: go back to the definition of `sort`.

EXERCISE 2 (finding all solutions)[a]

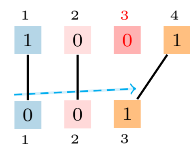
Give all the solutions to the constraint:

$$\begin{cases} X_1 \in [2,4], \quad X_2 \in [2,3], \quad X_3 \in [0,5], \quad X_4 \in [6,8], \quad X_5 \in [3,6], \\ Y_1 \in [3,4], \quad Y_2 \in [2,3], \quad Y_3 \in [0,5], \quad Y_4 \in [6,8], \quad Y_5 \in [3,6], \\ \texttt{sort}\begin{pmatrix} \langle X_1, & X_2, & X_3, & X_4, & X_5\rangle, \\ \langle Y_1, & Y_2, & Y_3, & Y_4, & Y_5\rangle \end{pmatrix}. \end{cases}$$

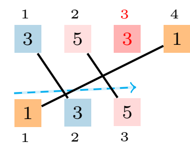
[a]Hint: first filter the bounds of the variables of the second argument wrt the chain of precedences; second, since the second argument can be computed from the first one, focus on the variables of the first argument and enumerate solutions in lexicographic order.

SOLUTION TO EXERCISE 1

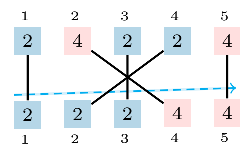
A. *No, since* $\langle 1,0,0,1\rangle$ *and* $\langle 0,0,1\rangle$ *do not have the same number of elements.*



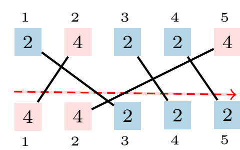
B. *No, since* $\langle 3,5,3,1\rangle$ *and* $\langle 1,3,5\rangle$ *do not have the same number of elements.*



C. *Yes, since* $\langle 2,2,2,4,4\rangle$ *is a permutation of* $\langle 2,4,2,2,4\rangle$ *and since the elements* $2,2,2,4,4$ *are sorted in non-decreasing order.*



D. *No, since the elements of* $\langle 4,4,2,2,2\rangle$ *are not sorted in non-decreasing order.*



SOLUTION TO EXERCISE 2

the four solutions

$\langle X_1, X_2, X_3, X_4, X_5\rangle, \langle Y_1, Y_2, Y_3, Y_4, Y_5\rangle$

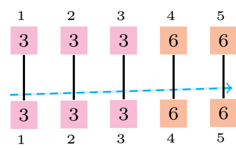
① $(\langle 3, 3, 3, 6, 6\rangle,\ \langle 3, 3, 3, 6, 6\rangle)$

② $(\langle 3, 3, 4, 6, 6\rangle,\ \langle 3, 3, 4, 6, 6\rangle)$

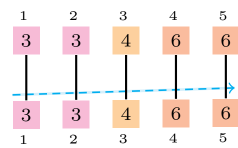
③ $(\langle 3, 3, 5, 6, 6\rangle,\ \langle 3, 3, 5, 6, 6\rangle)$

④ $(\langle 4, 3, 3, 6, 6\rangle,\ \langle 3, 3, 4, 6, 6\rangle)$

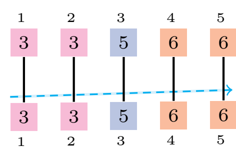
solution ①



solution ②



solution ③



solution ④