5.412 used_by

DESCRIPTION LINKS GRAPH AUTOMATON

Origin N. Beldiceanu

Constraint `used_by(VARIABLES1, VARIABLES2)`

Arguments

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

Restrictions

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

Purpose All the values of the variables of collection `VARIABLES2` are used by the variables of collection `VARIABLES1`.

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

The `used_by` constraint holds since, for each value occurring within the collection `VARIABLES2` $= \langle 1, 1, 2, 5\rangle$, its number of occurrences within `VARIABLES1` $= \langle 1, 9, 1, 5, 2, 1\rangle$ is greater than or equal to its number of occurrences within `VARIABLES2`:

- Value 1 occurs 3 times within $\langle 1, 9, 1, 5, 2, 1\rangle$ and 2 times within $\langle 1, 1, 2, 5\rangle$.
- Value 2 occurs 1 times within $\langle 1, 9, 1, 5, 2, 1\rangle$ and 1 times within $\langle 1, 1, 2, 5\rangle$.
- Value 5 occurs 1 times within $\langle 1, 9, 1, 5, 2, 1\rangle$ and 1 times within $\langle 1, 1, 2, 5\rangle$.

All solutions Figure 5.782 gives all solutions to the following non ground instance of the `used_by` constraint: $U_1 \in \{1, 5\}$, $U_2 \in [1, 2]$, $U_3 \in [1, 2]$, $V_1 \in [0, 2]$, $V_2 \in [2, 4]$, `used_by`$(\langle U_1, U_2, U_3\rangle, \langle V_1, V_2\rangle)$.

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

Figure 5.782: All solutions corresponding to the non ground example of the `used_by` constraint of the **All solutions** slot where identical values are coloured in the same way in both collections

Typical

$|\mathtt{VARIABLES1}| > 1$
range($\mathtt{VARIABLES1.var}$) > 1
$|\mathtt{VARIABLES2}| > 1$
range($\mathtt{VARIABLES2.var}$) > 1

Symmetries

- Items of VARIABLES1 are permutable.
- Items of VARIABLES2 are permutable.
- All occurrences of two distinct values in VARIABLES1.var or VARIABLES2.var can be swapped; all occurrences of a value in VARIABLES1.var or VARIABLES2.var can be renamed to any unused value.

Arg. properties

- Contractible wrt. VARIABLES2.
- Extensible wrt. VARIABLES1.
- Aggregate: VARIABLES1(union), VARIABLES2(union).

Algorithm

As described in [47] we can pad VARIABLES2 with dummy variables such that its cardinality will be equal to that cardinality of VARIABLES1. The domain of a dummy variable contains all of the values. Then, we have a same constraint between the two sets. Direct arc-consistency and bound-consistency algorithms based on a flow model are also proposed in [47, 49, 231]. The leftmost part of Figure 3.31 illustrates this flow model.

More recently [129, 130] presents a second filtering algorithm also achieving arc-consistency based on a mapping of the solutions to the used_by constraint to var-perfect matchings[16] in a bipartite intersection graph derived from the domain of the variables of the constraint in the following way. To each variable of the VARIABLES1 and VARIABLES2 collection corresponds a vertex of the intersection graph. There is an edge between a vertex associated with a variable of the VARIABLES1 collection and a vertex associated with a variable of the VARIABLES2 collection if and only if the corresponding variables have at least one value in common in their domains.

Reformulation

The used_by($\langle \mathtt{var}-U_1\ \mathtt{var}-U_2, \ldots, \mathtt{var}-U_{|\mathtt{VARIABLES1}|}\rangle, \langle \mathtt{var}-V_1\ \mathtt{var}-V_2, \ldots, \mathtt{var}-V_{|\mathtt{VARIABLES2}|}\rangle$) constraint can be expressed in term of a conjunction of $|\mathtt{VARIABLES2}|$ reified constraints of the form:

$$\sum_{1\leq j\leq |\mathtt{VARIABLES1}|}(V_i = U_j) \geq \sum_{1\leq j\leq |\mathtt{VARIABLES2}|}(V_i = V_j)\ (i \in [1, |\mathtt{VARIABLES2}|]).$$

Used in

int_value_precede_chain, k_used_by.

See also

generalisation: used_by_interval *(*variable *replaced by* variable/constant*)*, used_by_modulo *(*variable *replaced by* variable mod constant*)*, used_by_partition *(*variable *replaced by* variable $\in$ partition*)*.

implied by: same.

implies: uses.

soft variant: soft_used_by_var *(variable-based violation measure)*.

system of constraints: k_used_by.

[16] A *var-perfect matching* is a maximum matching covering all vertices corresponding to the variables of VARIABLES2.

Keywords

characteristic of a constraint: sort based reformulation, automaton, automaton with array of counters.

combinatorial object: multiset.

constraint arguments: constraint between two collections of variables.

filtering: flow, bipartite matching, arc-consistency, bound-consistency, DFS-bottleneck.

modelling: inclusion.

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** $\geq$ **NSINK** • **NSINK** $=$ |VARIABLES2|

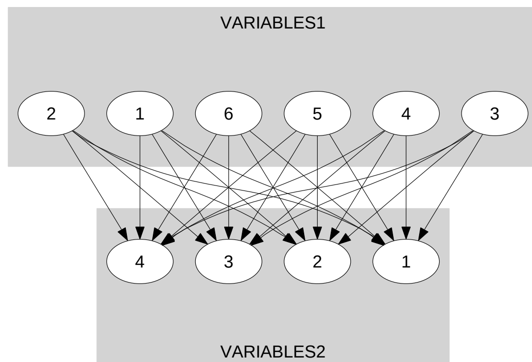
Graph model

Parts (A) and (B) of Figure 5.783 respectively show the initial and final graph associated with the **Example** slot. Since we use the **NSOURCE** and **NSINK** graph properties, the source and sink vertices of the 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. Each of them corresponds to an equivalence class according to the arc constraint. Note that the vertex corresponding to the variable assigned to value 9 was removed from the final graph since there is no arc for which the associated equality constraint holds. The used_by constraint holds since:

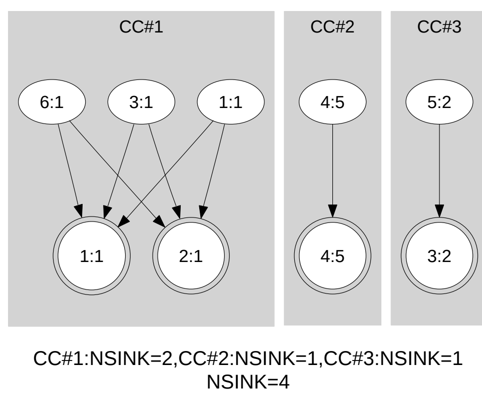
- For each connected component of the final graph the number of sources is greater than or equal to the number of sinks.
- The number of sinks of the final graph is equal to |VARIABLES2|.

Signature

Since the initial graph contains only sources and sinks, and since sources of the initial graph cannot become sinks of the final graph, we have that the maximum number of sinks of the final graph is equal to |VARIABLES2|. Therefore we can rewrite **NSINK** $=$ |VARIABLES2| to **NSINK** $\geq$ |VARIABLES2| and simplify $\underline{\overline{\mathbf{NSINK}}}$ to $\overline{\mathbf{NSINK}}$.



(A)



(B)

Figure 5.783: Initial and final graph of the `used_by` constraint

Automaton Figure 5.784 depicts the automaton associated with the `used_by` constraint. To each item of the collection `VARIABLES1` corresponds a signature variable S_i that is equal to 0. To each item of the collection `VARIABLES2` corresponds a signature variable $S_{i+|\texttt{VARIABLES1}|}$ that is equal to 1.

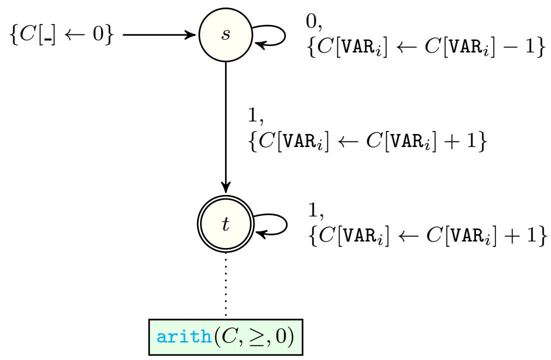


Figure 5.784: Automaton of the `used_by` constraint