5.414 used_by_modulo

DESCRIPTION LINKS GRAPH

Origin

Derived from used_by.

Constraint

used_by_modulo(VARIABLES1, VARIABLES2, M)

Arguments

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

Restrictions

|VARIABLES1| ≥ |VARIABLES2|
required(VARIABLES1, var)
required(VARIABLES2, var)
M > 0

Purpose

For each integer R in $[0, \mathtt{M}-1]$, let $N1_R$ (respectively $N2_R$) denote the number of variables of VARIABLES1 (respectively VARIABLES2) that have R as a rest when divided by M. For all R in $[0, \mathtt{M}-1]$ we have $N2_R > 0 \Rightarrow N1_R \geq N2_R$.

Example

$(\langle 1, 9, 4, 5, 2, 1\rangle, \langle 7, 1, 2, 5\rangle, 3)$

The values of the collection VARIABLES2 $= \langle 7, 1, 2, 5\rangle$ are respectively associated with the equivalence classes $7 \bmod 3 = 1$, $1 \bmod 3 = 1$, $2 \bmod 3 = 2$, $5 \bmod 3 = 2$. Therefore the equivalence classes 1 and 2 are respectively used 2 and 2 times.

Similarly, the values of the collection VARIABLES1 $= \langle 1, 9, 4, 5, 2, 1\rangle$ associated with the equivalence classes $1 \bmod 3 = 1$, $9 \bmod 3 = 0$, $4 \bmod 3 = 1$, $5 \bmod 3 = 2$, $2 \bmod 3 = 2$, $1 \bmod 3 = 1$. Therefore the equivalence classes 0, 1 and 2 are respectively used 1, 3 and 2 times.

Consequently, the used_by_modulo constraint holds since, for each equivalence class associated with the collection VARIABLES2 $= \langle 7, 1, 2, 5\rangle$, its number of occurrences within VARIABLES1 $= \langle 1, 9, 4, 5, 2, 1\rangle$ is greater than or equal to its number of occurrences within VARIABLES2:

- The equivalence class 1 occurs 3 times within $\langle 1, 9, 4, 5, 2, 1\rangle$ and 2 times within $\langle 7, 1, 2, 5\rangle$.
- The equivalence class 2 occurs 2 times within $\langle 1, 9, 4, 5, 2, 1\rangle$ and 2 times within $\langle 7, 1, 2, 5\rangle$.

Typical

|VARIABLES1| > 1
range(VARIABLES1.var) > 1
|VARIABLES2| > 1
range(VARIABLES2.var) > 1
M > 1
M <maxval(VARIABLES1.var)
M <maxval(VARIABLES2.var)

Symmetries

- Items of VARIABLES1 are permutable.
- Items of VARIABLES2 are permutable.
- An occurrence of a value u of VARIABLES1.var can be replaced by any other value v such that v is congruent to u modulo M.
- An occurrence of a value u of VARIABLES2.var can be replaced by any other value v such that v is congruent to u modulo M.

Arg. properties

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

Used in k_used_by_modulo.

See also

implied by: same_modulo.

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

specialisation: used_by *(*variable mod constant *replaced by* variable*)*.

system of constraints: k_used_by_modulo.

Keywords

characteristic of a constraint: modulo, sort based reformulation.

constraint arguments: constraint between two collections of variables.

modelling: inclusion.

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

Graph model

Parts (A) and (B) of Figure 5.786 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 that takes value 9 was removed from the final graph since there is no arc for which the associated equivalence constraint holds. The `used_by_modulo` 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** ≥ |`VARIABLES2`| and simplify $\underline{\overline{\mathbf{NSINK}}}$ to $\overline{\mathbf{NSINK}}$.

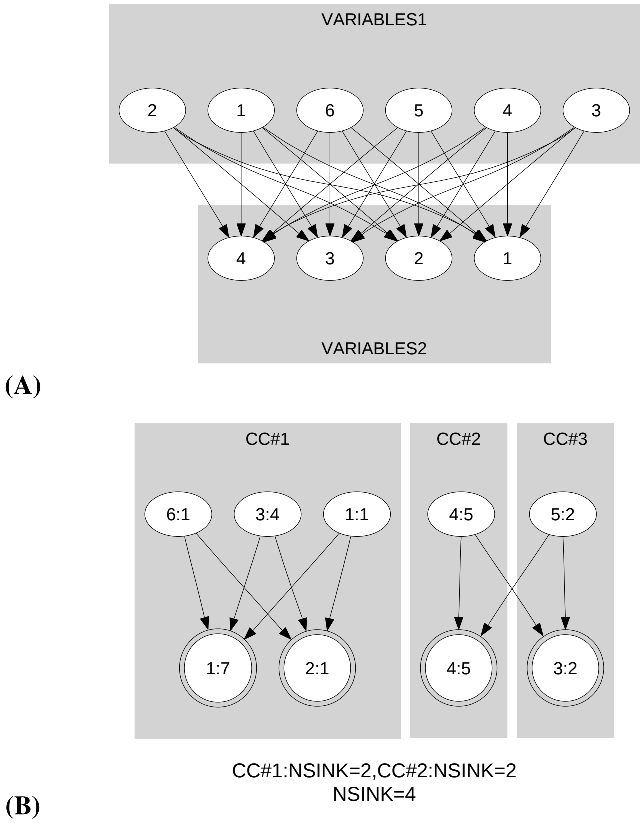

Figure 5.786: Initial and final graph of the `used_by_modulo` constraint