5.224 lex_chain_greatereq

DESCRIPTION LINKS GRAPH

Origin Derived from lex_chain_lesseq

Constraint lex_chain_greatereq(VECTORS)

Usual name lex_chain

Type VECTOR : collection(var−dvar)

Argument VECTORS : collection(vec − VECTOR)

Restrictions
$|\text{VECTOR}| \geq 1$
required(VECTOR, var)
required(VECTORS, vec)
same_size(VECTORS, vec)

Purpose For each pair of consecutive vectors VECTOR_i and VECTOR_{i+1} of the VECTORS collection we have that VECTOR_i is lexicographically greater than or equal to VECTOR_{i+1}. Given two vectors, $\vec{X}$ and $\vec{Y}$ of n components, $\langle X_0, \ldots, X_{n-1}\rangle$ and $\langle Y_0, \ldots, Y_{n-1}\rangle$, $\vec{X}$ is *lexicographically greater than or equal to* $\vec{Y}$ if and only if $n = 0$ or $X_0 > Y_0$ or $X_0 = Y_0$ and $\langle X_1, \ldots, X_{n-1}\rangle$ is lexicographically greater than or equal to $\langle Y_1, \ldots, Y_{n-1}\rangle$.

Example $(\langle \text{vec} - \langle 5,2,6,2\rangle, \text{vec} - \langle 5,2,6,2\rangle, \text{vec} - \langle 5,2,3,9\rangle\rangle)$

The lex_chain_greatereq constraint holds since:

- The first vector $\langle 5,2,6,2\rangle$ of the VECTORS collection is lexicographically greater than or equal to the second vector $\langle 5,2,6,2\rangle$ of the VECTORS collection.
- The second vector $\langle 5,2,6,2\rangle$ of the VECTORS collection is lexicographically greater than or equal to the third vector $\langle 5,2,3,9\rangle$ of the VECTORS collection.

Typical
$|\text{VECTOR}| > 1$
$|\text{VECTORS}| > 1$

Arg. properties

- Contractible wrt. VECTORS.
- Suffix-contractible wrt. VECTORS.vec (*remove items from same position*).

Usage This constraint was motivated for breaking symmetry: more precisely when one wants to lexicographically order the consecutive columns of a matrix of decision variables. A further motivation is that using a set of lexicographic ordering constraints between two vectors does usually not allows to come up with a complete pruning.

Algorithm

A filtering algorithm achieving arc-consistency for a chain of lexicographical ordering constraints is presented in [95].

Six different ways of integrating a chain of lexicographical ordering constraints within non-overlapping constraints like `diffn` or `geost` and within their corresponding necessary condition like the `cumulative` constraint are shown in [3].

See also

common keyword: `lex_between`, `lex_greater`, `lex_less`, `lex_lesseq` *(lexicographic order)*.

implied by: `lex_chain_greater` *(non-strict order implied by strict order)*.

part of system of constraints: `lex_greatereq`.

used in graph description: `lex_greatereq`.

Keywords

characteristic of a constraint: vector.

constraint type: system of constraints, decomposition, order constraint.

filtering: arc-consistency.

heuristics: heuristics and lexicographical ordering.

symmetry: symmetry, matrix symmetry, lexicographic order.

Arc input(s)	VECTORS
Arc generator	*PATH* $\mapsto$ collection(vectors1, vectors2)
Arc arity	2
Arc constraint(s)	lex_lesseq(vectors1.vec, vectors2.vec)
Graph property(ies)	**NARC** = |VECTORS| − 1

Graph model Parts (A) and (B) of Figure 5.479 respectively show the initial and final graph associated with the **Example** slot. Since we use the **NARC** graph property, the arcs of the final graph are stressed in bold. The lex_chain_greatereq constraint holds since all the arc constraints of the initial graph are satisfied.

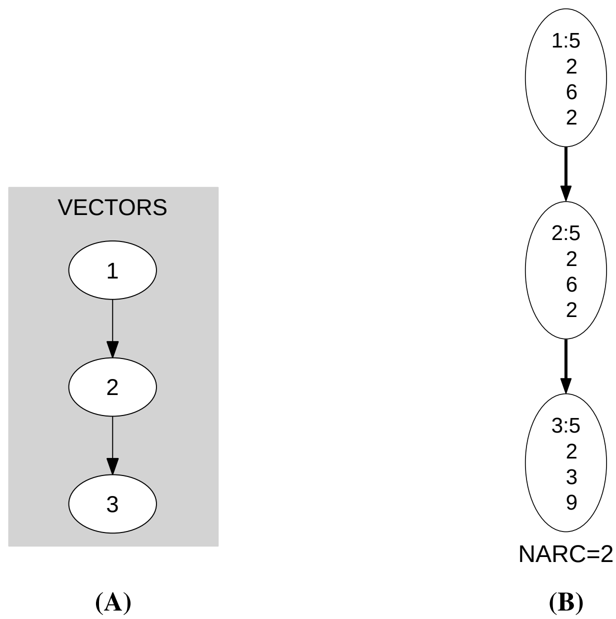

Figure 5.479: Initial and final graph of the lex_chain_greatereq constraint

Signature Since we use the *PATH* arc generator on the VECTORS collection the number of arcs of the initial graph is equal to |VECTORS| − 1. For this reason we can rewrite **NARC** = |VECTORS| − 1 to **NARC** $\geq$ |VECTORS| − 1 and simplify $\underline{\mathbf{NARC}}$ to $\overline{\mathbf{NARC}}$.