5.200 lex_lesseq

DESCRIPTION LINKS GRAPH AUTOMATON

Origin CHIP

Constraint `lex_lesseq(VECTOR1, VECTOR2)`

Synonyms `lexeq`, `lex_chain`, `rel`, `lesseq`, `leq`, `lex_leq`.

Arguments

```
VECTOR1 : collection(var-dvar)
VECTOR2 : collection(var-dvar)
```

Restrictions

```
required(VECTOR1, var)
required(VECTOR2, var)
|VECTOR1| = |VECTOR2|
```

Purpose

VECTOR1 is *lexicographically less than or equal to* VECTOR2. Given two vectors, $\vec{X}$ and $\vec{Y}$ of n components, $\langle X_0, \dots, X_{n-1}\rangle$ and $\langle Y_0, \dots, Y_{n-1}\rangle$, $\vec{X}$ is *lexicographically less than or equal to* $\vec{Y}$ if and only if $n = 0$ or $X_0 < Y_0$ or $X_0 = Y_0$ and $\langle X_1, \dots, X_{n-1}\rangle$ is *lexicographically less than or equal to* $\langle Y_1, \dots, Y_{n-1}\rangle$.

Example

$$\left(\begin{array}{l}\langle 5,2,3,1\rangle,\\ \langle 5,2,6,2\rangle\end{array}\right)$$
$$\left(\begin{array}{l}\langle 5,2,3,9\rangle,\\ \langle 5,2,3,9\rangle\end{array}\right)$$

The `lex_lesseq` constraints associated with the first and second examples hold since:

- Within the first example VECTOR1 $= \langle 5,2,3,1\rangle$ is lexicographically less than or equal to VECTOR2 $= \langle 5,2,6,2\rangle$.
- Within the second example VECTOR1 $= \langle 5,2,3,9\rangle$ is lexicographically less than or equal to VECTOR2 $= \langle 5,2,3,9\rangle$.

Symmetries

- VECTOR1.var can be decreased.
- VECTOR2.var can be increased.

Remark A *multiset ordering* constraint and its corresponding filtering algorithm are described in [154].

Algorithm The first filtering algorithm maintaining arc-consistency for this constraint was presented in [153]. A second filtering algorithm maintaining arc-consistency and detecting entailment in a more eager way, was given in [87]. This second algorithm was derived from a deterministic finite automata. A third filtering algorithm extending the algorithm presented in [153] detecting entailment is given in the PhD thesis of Z. Kızıltan [212, page 95]. The previous thesis [212, pages 105–109] presents also a filtering algorithm handling the fact that a given variable has more than one occurrence. Finally, T. Frühwirth shows how to encode lexicographic ordering constraints within the context of CHR [155] in [156].

Reformulation The following reformulations in term of arithmetic and/or logical expressions exist for enforcing the *lexicographically less than or equal to* constraint. The first one converts $\vec{X}$ and $\vec{Y}$ into numbers and post an inequality constraint. It assumes all components of $\vec{X}$ and $\vec{Y}$ to be within $[0, a-1]$:

$$a^{n-1}X_0 + a^{n-2}X_1 + \ldots + a^0 X_{n-1} \leq a^{n-1}Y_0 + a^{n-2}Y_1 + \ldots + a^0 Y_{n-1}$$

Since the previous reformulation can only be used with small values of n and a, W. Harvey came up with the following alternative model that maintains arc-consistency:

$$(X_0 < Y_0 + (X_1 < Y_1 + (\ldots + (X_{n-1} < Y_{n-1} + 1)\ldots))) = 1$$

Finally, the *lexicographically less than or equal to* constraint can be expressed as a conjunction or a disjunction of constraints:

$$\begin{array}{rl}
X_0 \leq Y_0 & \wedge \\
(X_0 = Y_0) \Rightarrow X_1 \leq Y_1 & \wedge \\
(X_0 = Y_0 \wedge X_1 = Y_1) \Rightarrow X_2 \leq Y_2 & \wedge \\
\vdots & \\
(X_0 = Y_0 \wedge X_1 = Y_1 \wedge \ldots \wedge X_{n-2} = Y_{n-2}) \Rightarrow X_{n-1} \leq Y_{n-1} &
\end{array}$$

$$\begin{array}{rl}
X_0 < Y_0 & \vee \\
X_0 = Y_0 \wedge X_1 < Y_1 & \vee \\
X_0 = Y_0 \wedge X_1 = Y_1 \wedge X_2 < Y_2 & \vee \\
\vdots & \\
X_0 = Y_0 \wedge X_1 = Y_1 \wedge \ldots \wedge X_{n-2} = Y_{n-2} \wedge X_{n-1} \leq Y_{n-1} &
\end{array}$$

When used separately, the two previous logical decompositions do not maintain arc-consistency.

Systems `lexEq` in **Choco**, `rel` in **Gecode**, `lex_chain` in **SICStus**.

Used in `lex_between`, `lex_chain_lesseq`, `ordered_atleast_nvector`, `ordered_atmost_nvector`, `ordered_nvector`.

See also **common keyword:** `allperm`, `cond_lex_lesseq` *(lexicographic order)*, `lex2` *(matrix symmetry,lexicographic order)*, `lex_chain_less` *(lexicographic order)*, `lex_different` *(vector)*, `strict_lex2` *(matrix symmetry,lexicographic order)*.

implied by: `lex_equal`, `lex_less`, `lex_lesseq_allperm`.

implies if swap arguments: `lex_greatereq`.

negation: `lex_greater`.

system of constraints: `lex_between`, `lex_chain_lesseq`.

Keywords **characteristic of a constraint:** vector, automaton, automaton without counters, reified automaton constraint, derived collection.

constraint network structure: Berge-acyclic constraint network.

constraint type: order constraint.

filtering: duplicated variables, arc-consistency.

heuristics: heuristics and lexicographical ordering.

symmetry: symmetry, matrix symmetry, lexicographic order, multiset ordering.

Derived Collections

$$\texttt{col}\left(\begin{array}{l}\texttt{DESTINATION}-\texttt{collection}(\texttt{index}-\texttt{int},\texttt{x}-\texttt{int},\texttt{y}-\texttt{int}),\\ [\texttt{item}(\texttt{index}-0,\texttt{x}-0,\texttt{y}-0)]\end{array}\right)$$

$$\texttt{col}\left(\begin{array}{l}\texttt{COMPONENTS}-\texttt{collection}(\texttt{index}-\texttt{int},\texttt{x}-\texttt{dvar},\texttt{y}-\texttt{dvar}),\\ \left[\ \texttt{item}(\texttt{index}-\texttt{VECTOR1.key},\texttt{x}-\texttt{VECTOR1.var},\texttt{y}-\texttt{VECTOR2.var})\ \right]\end{array}\right)$$

Arc input(s) COMPONENTS DESTINATION

Arc generator $PRODUCT(PATH, VOID) \mapsto \texttt{collection}(\texttt{item1}, \texttt{item2})$

Arc arity 2

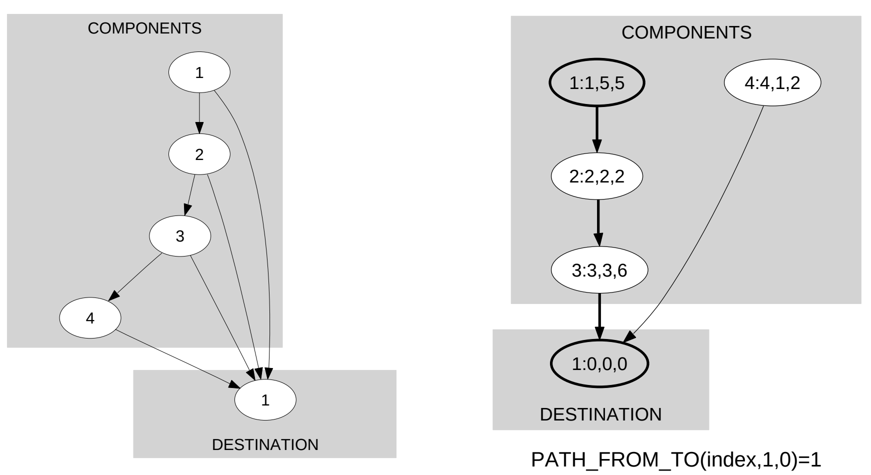
Arc constraint(s)

$$\bigvee\left(\begin{array}{l}\texttt{item2.index} > 0 \wedge \texttt{item1.x} = \texttt{item1.y},\\ \texttt{item1.index} < |\texttt{VECTOR1}| \wedge \texttt{item2.index} = 0 \wedge \texttt{item1.x} < \texttt{item1.y},\\ \texttt{item1.index} = |\texttt{VECTOR1}| \wedge \texttt{item2.index} = 0 \wedge \texttt{item1.x} \leq \texttt{item1.y}\end{array}\right)$$

Graph property(ies) **PATH_FROM_TO**$(\texttt{index}, 1, 0) = 1$

Graph model Parts (A) and (B) of Figure 5.392 respectively show the initial and final graph associated with the first example of the **Example** slot. Since we use the **PATH_FROM_TO** graph property we show on the final graph the following information:

- The vertices, which respectively correspond to the start and the end of the required path, are stressed in bold.
- The arcs on the required path are also stressed in bold.



(A) **(B)**

Figure 5.392: Initial and final graph of the `lex_lesseq` constraint

The vertices of the initial graph are generated in the following way:

- We create a vertex c_i for each pair of components that both have the same index i.
- We create an additional dummy vertex called d.

The arcs of the initial graph are generated in the following way:

- We create an arc between c_i and d. When c_i was generated from the last components of both vectors We associate to this arc the arc constraint $\texttt{item}_1.x \leq \texttt{item}_2.y$; Otherwise we associate to this arc the arc constraint $\texttt{item}_1.x < \texttt{item}_2.y$;
- We create an arc between c_i and c_{i+1}. We associate to this arc the arc constraint $\texttt{item}_1.x = \texttt{item}_2.y$.

The `lex_lesseq` constraint holds when there exist a path from c_1 to d. This path can be interpreted as a maximum sequence of *equality* constraints on the prefix of both vectors, possibly followed by a *less than* constraint.

Signature

Since the maximum value returned by the graph property $\mathbf{PATH\_FROM\_TO}$ is equal to 1 we can rewrite $\mathbf{PATH\_FROM\_TO}(\texttt{index}, 1, 0) = 1$ to $\mathbf{PATH\_FROM\_TO}(\texttt{index}, 1, 0) \geq 1$. Therefore we simplify $\underline{\overline{\mathbf{PATH\_FROM\_TO}}}$ to $\overline{\mathbf{PATH\_FROM\_TO}}$.

Automaton Figure 5.393 depicts the automaton associated with the lex_lesseq constraint. Let $\mathtt{VAR1}_i$ and $\mathtt{VAR2}_i$ respectively be the var attributes of the i^{th} items of the VECTOR1 and the VECTOR2 collections. To each pair $(\mathtt{VAR1}_i, \mathtt{VAR2}_i)$ corresponds a signature variable $\mathtt{S}_i$ as well as the following signature constraint: $(\mathtt{VAR1}_i < \mathtt{VAR2}_i \Leftrightarrow \mathtt{S}_i = 1) \wedge (\mathtt{VAR1}_i = \mathtt{VAR2}_i \Leftrightarrow \mathtt{S}_i = 2) \wedge (\mathtt{VAR1}_i > \mathtt{VAR2}_i \Leftrightarrow \mathtt{S}_i = 3)$.

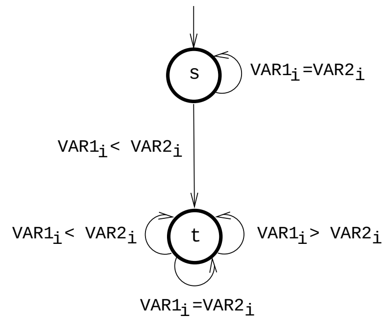


Figure 5.393: Automaton of the lex_lesseq constraint

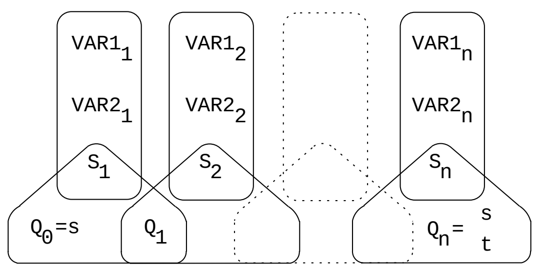


Figure 5.394: Hypergraph of the reformulation corresponding to the automaton of the lex_lesseq constraint