5.223 lex_between

DESCRIPTION LINKS AUTOMATON

Origin [95]

Constraint lex_between(LOWER_BOUND, VECTOR, UPPER_BOUND)

Synonym between.

Arguments

LOWER_BOUND	:	collection(var−int)
VECTOR	:	collection(var−dvar)
UPPER_BOUND	:	collection(var−int)

Restrictions

required(LOWER_BOUND, var)
required(VECTOR, var)
required(UPPER_BOUND, var)
|LOWER_BOUND| = |VECTOR|
|UPPER_BOUND| = |VECTOR|
lex_lesseq(LOWER_BOUND, VECTOR)
lex_lesseq(VECTOR, UPPER_BOUND)

Purpose

The vector VECTOR is lexicographically greater than or equal to the fixed vector LOWER_BOUND and lexicographically smaller than or equal to the fixed vector UPPER_BOUND.

Example

$(\langle 5,2,3,9\rangle, \langle 5,2,6,2\rangle, \langle 5,2,6,3\rangle)$

The lex_between constraint holds since:

- The vector VECTOR $= \langle 5,2,6,2\rangle$ is greater than or equal to the vector LOWER_BOUND $= \langle 5,2,3,9\rangle$.
- The vector VECTOR $= \langle 5,2,6,2\rangle$ is less than or equal to the vector UPPER_BOUND $= \langle 5,2,6,3\rangle$.

Typical

|LOWER_BOUND| > 1
lex_lesseq(LOWER_BOUND, UPPER_BOUND)

Symmetries

- LOWER_BOUND.var can be decreased.
- UPPER_BOUND.var can be increased.

Arg. properties

Suffix-contractible wrt. LOWER_BOUND, VECTOR and UPPER_BOUND (*remove items from same position*).

Usage This constraint does usually not occur explicitly in practice. However it shows up indirectly in the context of the lex_chain_less and the lex_chain_lesseq constraints: in order to have a complete filtering algorithm for the lex_chain_less and the lex_chain_lesseq constraints one has to come up with a complete filtering algorithm for the lex_between constraint. The reason is that the lex_chain_less as well as the lex_chain_lesseq constraints both compute feasible lower and upper bounds for each vector they mention. Therefore one ends up with a lex_between constraint for each vector of the lex_chain_less and lex_chain_lesseq constraints.

Algorithm [95].

Reformulation The lex_between(LOWER_BOUND, VECTORS, UPPER_BOUND) constraint can be expressed as the conjunction lex_lesseq(LOWER_BOUND, VECTORS) $\wedge$ lex_lesseq(VECTORS, UPPER_BOUND).

Systems lexChainEq in **Choco**, lex_chain in **SICStus**.

See also **common keyword:** lex_chain_greater, lex_chain_greatereq, lex_chain_less, lex_chain_lesseq, lex_greater, lex_greatereq, lex_less *(lexicographic order)*.

part of system of constraints: lex_lesseq.

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

constraint network structure: Berge-acyclic constraint network.

constraint type: order constraint, system of constraints.

filtering: arc-consistency.

symmetry: symmetry, lexicographic order.

Automaton Figure 5.495 depicts the automaton associated with the `lex_between` constraint. Let $\mathtt{L}_i$, $\mathtt{V}_i$ and $\mathtt{U}_i$ respectively be the `var` attributes of the i^{th} items of the `LOWER_BOUND`, the `VECTOR` and the `UPPER_BOUND` collections. To each triple $(\mathtt{L}_i, \mathtt{V}_i, \mathtt{U}_i)$ corresponds a signature variable S_i as well as the following signature constraint:

$$(\mathtt{L}_i < \mathtt{V}_i) \wedge (\mathtt{V}_i < \mathtt{U}_i) \Leftrightarrow S_i = 0 \wedge$$
$$(\mathtt{L}_i < \mathtt{V}_i) \wedge (\mathtt{V}_i = \mathtt{U}_i) \Leftrightarrow S_i = 1 \wedge$$
$$(\mathtt{L}_i < \mathtt{V}_i) \wedge (\mathtt{V}_i > \mathtt{U}_i) \Leftrightarrow S_i = 2 \wedge$$
$$(\mathtt{L}_i = \mathtt{V}_i) \wedge (\mathtt{V}_i < \mathtt{U}_i) \Leftrightarrow S_i = 3 \wedge$$
$$(\mathtt{L}_i = \mathtt{V}_i) \wedge (\mathtt{V}_i = \mathtt{U}_i) \Leftrightarrow S_i = 4 \wedge$$
$$(\mathtt{L}_i = \mathtt{V}_i) \wedge (\mathtt{V}_i > \mathtt{U}_i) \Leftrightarrow S_i = 5 \wedge$$
$$(\mathtt{L}_i > \mathtt{V}_i) \wedge (\mathtt{V}_i < \mathtt{U}_i) \Leftrightarrow S_i = 6 \wedge$$
$$(\mathtt{L}_i > \mathtt{V}_i) \wedge (\mathtt{V}_i = \mathtt{U}_i) \Leftrightarrow S_i = 7 \wedge$$
$$(\mathtt{L}_i > \mathtt{V}_i) \wedge (\mathtt{V}_i > \mathtt{U}_i) \Leftrightarrow S_i = 8.$$

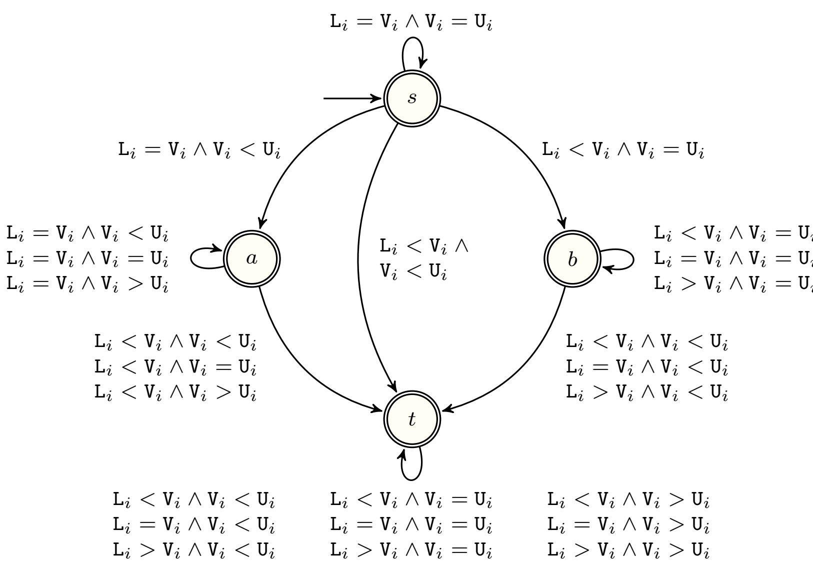

Figure 5.495: Automaton of the `lex_between` constraint

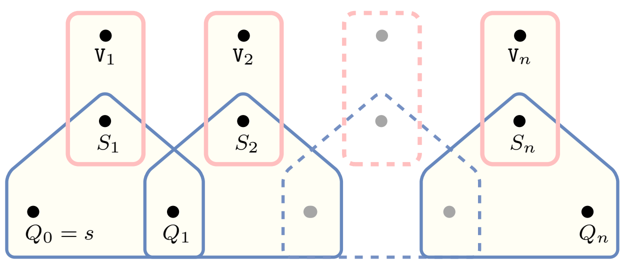

Figure 5.496: Hypergraph of the reformulation corresponding to the automaton of the `lex_between` constraint (since all states of the automaton are accepting there is no restriction on the last variable Q_n)