

5.81 cond_lex_greater

	DESCRIPTION	LINKS	AUTOMATON
Origin	Inspired by [437].		
Constraint	<code>cond_lex_greater(VECTOR1, VECTOR2, PREFERENCE_TABLE)</code>		
Type	TUPLE_OF_VALS : <code>collection(val-int)</code>		
Arguments	VECTOR1 : <code>collection(var-dvar)</code> VECTOR2 : <code>collection(var-dvar)</code> PREFERENCE_TABLE : <code>collection(tuple - TUPLE_OF_VALS)</code>		
Restrictions	$ TUPLE_OF_VALS \geq 1$ <code>required(TUPLE_OF_VALS, val)</code> <code>required(VECTOR1, var)</code> <code>required(VECTOR2, var)</code> $ VECTOR1 = VECTOR2 $ $ VECTOR1 = TUPLE_OF_VALS $ <code>required(PREFERENCE_TABLE, tuple)</code> <code>same_size(PREFERENCE_TABLE, tuple)</code> <code>distinct(PREFERENCE_TABLE, [])</code> <code>in_relation(VECTOR1, PREFERENCE_TABLE)</code> <code>in_relation(VECTOR2, PREFERENCE_TABLE)</code>		
Purpose	<div style="border: 1px solid pink; padding: 5px;"> VECTOR1 and VECTOR2 are both assigned to the I^{th} and J^{th} items of the collection PREFERENCE_TABLE such that $I > J$. </div>		
Example	<div style="border: 1px solid blue; padding: 10px; display: inline-block;"> $\left(\begin{array}{l} \langle 0, 0 \rangle, \\ \langle 1, 0 \rangle, \\ \langle \text{tuple} - \langle 1, 0 \rangle, \\ \text{tuple} - \langle 0, 1 \rangle, \\ \text{tuple} - \langle 0, 0 \rangle, \\ \text{tuple} - \langle 1, 1 \rangle \end{array} \right)$ </div> <p>The <code>cond_lex_greater</code> constraint holds since VECTOR1 and VECTOR2 are respectively assigned to the third and first items of the collection PREFERENCE_TABLE.</p>		
Typical	$ TUPLE_OF_VALS > 1$ $ VECTOR1 > 1$ $ VECTOR2 > 1$ $ PREFERENCE_TABLE > 1$		

Symmetries

- Items of `VECTOR1`, `VECTOR2` and `PREFERENCE_TABLE.tuple` are [permutable](#) (*same permutation used*).
- All occurrences of two distinct tuples of values in `VECTOR1`, `VECTOR2` or `PREFERENCE_TABLE.tuple` can be [swapped](#); all occurrences of a tuple of values in `VECTOR1`, `VECTOR2` or `PREFERENCE_TABLE.tuple` can be [renamed](#) to any unused tuple of values.

Usage

See [cond_lex_cost](#).

See also

common keyword: [cond_lex_cost](#), [cond_lex_greatereq](#), [cond_lex_less](#), [cond_lex_lesseq](#) (*preferences*), [lex_greater](#) (*lexicographic order*).

implies: [cond_lex_greatereq](#).

Keywords

characteristic of a constraint: vector, automaton.

constraint network structure: Berge-acyclic constraint network.

constraint type: order constraint.

filtering: arc-consistency.

modelling: preferences.

symmetry: lexicographic order.

Automaton

Figure 5.187 depicts the automaton associated with the preference table of the `cond_lex_greater` constraint given in the example. Let $VAR1_k$ and $VAR2_k$ respectively be the `var` attributes of the k^{th} items of the `VECTOR1` and the `VECTOR2` collections. Figure 5.188 depicts the reformulation of the `cond_lex_greater` constraint. This reformulation uses:

- Two occurrences of the automaton depicted by Figure 5.187 for computing the positions I and J within the preference table corresponding to `VECTOR1` and `VECTOR2`.
- The binary constraint $I > J$.

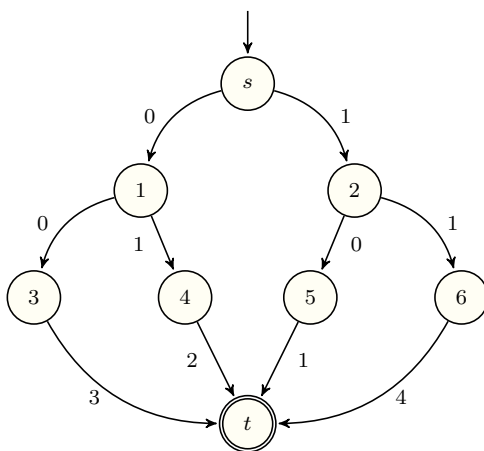


Figure 5.187: Automaton associated with the preference table of the `cond_lex_greater` constraint given in the **Example** slot

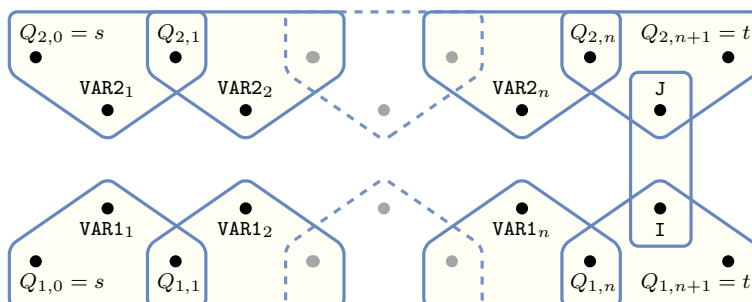


Figure 5.188: Hypergraph of the reformulation corresponding to the `cond_lex_greater` constraint: it uses two occurrences of the automaton of Figure 5.187 and the constraint $I > J$

20060430

869