

## 5.66 cond\_lex\_cost

	DESCRIPTION	LINKS	AUTOMATON
<b>Origin</b>	Inspired by [398].		
<b>Constraint</b>	<code>cond_lex_cost(VECTOR, PREFERENCE_TABLE, COST)</code>		
<b>Type</b>	TUPLE_OF_VALS : <code>collection(val-int)</code>		
<b>Arguments</b>	VECTOR : <code>collection(var-dvar)</code> PREFERENCE_TABLE : <code>collection(tuple - TUPLE_OF_VALS)</code> COST : <code>dvar</code>		
<b>Restrictions</b>	<code>required(TUPLE_OF_VALS, val)</code> <code>required(VECTOR, var)</code> $ \text{VECTOR}  =  \text{TUPLE\_OF\_VALS} $ <code>required(PREFERENCE_TABLE, tuple)</code> <code>same_size(PREFERENCE_TABLE, tuple)</code> <code>distinct(PREFERENCE_TABLE, [])</code> <code>in_relation(VECTOR, PREFERENCE_TABLE)</code> $\text{COST} \geq 1$ $\text{COST} \leq  \text{PREFERENCE\_TABLE} $		
<b>Purpose</b>	VECTOR is assigned to the $\text{COST}^{\text{th}}$ item of the collection PREFERENCE_TABLE.		
<b>Example</b>	$\left( \begin{array}{l} \langle 0, 1 \rangle, \\ \text{tuple} - \langle 1, 0 \rangle, \\ \langle \text{tuple} - \langle 0, 1 \rangle, \\ \text{tuple} - \langle 0, 0 \rangle, \rangle, 2 \\ \text{tuple} - \langle 1, 1 \rangle \end{array} \right)$		
	The <code>cond_lex_cost</code> constraint holds since VECTOR is assigned to the second item of the collection PREFERENCE_TABLE.		
<b>Typical</b>	$ \text{TUPLE\_OF\_VALS}  > 1$ $ \text{VECTOR}  > 1$ $ \text{PREFERENCE\_TABLE}  > 1$		
<b>Symmetries</b>	<ul style="list-style-type: none"> <li>Items of VECTOR and PREFERENCE_TABLE.<code>tuple</code> are <b>permutable</b> (<i>same permutation used</i>).</li> <li>All occurrences of two distinct tuples of values in VECTOR or PREFERENCE_TABLE.<code>tuple</code> can be <b>swapped</b>; all occurrences of a tuple of values in VECTOR or PREFERENCE_TABLE.<code>tuple</code> can be <b>renamed</b> to any unused tuple of values.</li> </ul>		

**Usage**

We consider an example taken from [398] where a customer has to decide among vacations. There are two seasons when he can travel (`spring` and `summer`) and two locations `Naples` and `Helsinki`. Furthermore assume that location is more important than season and the preferred period of the year depends on the selected location. The travel preferences of a customer are explicitly defined by stating the preferences ordering among the possible tuples of values  $\langle \text{Naples}, \text{spring} \rangle$ ,  $\langle \text{Naples}, \text{summer} \rangle$ ,  $\langle \text{Helsinki}, \text{spring} \rangle$  and  $\langle \text{Helsinki}, \text{summer} \rangle$ . For instance we may state within the preference table `PREFERENCE_TABLE` of the `cond_lex_cost` constraint the preference ordering  $\langle \text{Naples}, \text{spring} \rangle \succ \langle \text{Helsinki}, \text{summer} \rangle \succ \langle \text{Helsinki}, \text{spring} \rangle \succ \langle \text{Naples}, \text{summer} \rangle$ , which denotes the fact that our customer prefers `Naples` in the `spring` and `Helsinki` in the `summer`, and a vacation in `spring` is preferred over `summer`. Finally a solution minimising the cost variable `COST` will match the preferences stated by our customer.

**See also**

**attached to cost variant:** `in_relation` (*COST parameter removed*).

**common keyword:** `cond_lex_greater`, `cond_lex_greatereq`, `cond_lex_less`, `cond_lex_lesseq` (*preferences*).

**specialisation:** `element` (*tuple of variables replaced by single variable*).

**Keywords**

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

**constraint network structure:** `Berge-acyclic constraint network`.

**constraint type:** `order constraint`.

**filtering:** `arc-consistency`, `cost filtering constraint`.

**modelling:** `preferences`.

**symmetry:** `lexicographic order`.

**Automaton**

Figure 5.137 depicts the automaton associated with `cond_lex_lesseq` constraint. Let  $\text{VAR}_k$  denote the  $\text{var}$  attribute of the  $k^{\text{th}}$  item of the `VECTOR` collection. Figure 5.138 depicts the reformulation of the `cond_lex_cost` constraint.

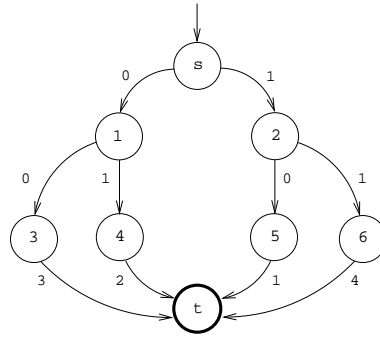


Figure 5.137: Automaton of the `cond_lex_cost` constraint given in the example

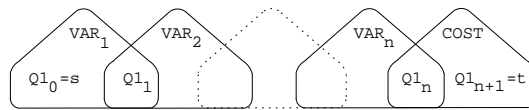


Figure 5.138: Hypergraph of the reformulation corresponding to the automaton of the `cond_lex_cost` constraint

20060416

657