5.80 cond_lex_cost

DESCRIPTION LINKS AUTOMATON

Origin Inspired by [437].

Constraint cond_lex_cost(VECTOR, PREFERENCE_TABLE, COST)

Type

TUPLE_OF_VALS	:	collection(val−int)

Arguments

VECTOR	:	collection(var−dvar)
PREFERENCE_TABLE	:	collection(tuple − TUPLE_OF_VALS)
COST	:	dvar

Restrictions

|TUPLE_OF_VALS| ≥ 1
required(TUPLE_OF_VALS, val)
required(VECTOR, var)
|VECTOR| = |TUPLE_OF_VALS|
required(PREFERENCE_TABLE, tuple)
same_size(PREFERENCE_TABLE, tuple)
distinct(PREFERENCE_TABLE, [])
in_relation(VECTOR, PREFERENCE_TABLE)
COST ≥ 1
COST ≤ |PREFERENCE_TABLE|

Purpose VECTOR is assigned to the $\texttt{COST}^{th}$ item of the collection PREFERENCE_TABLE.

Example

$$\left(\begin{array}{l}\langle 0,1\rangle,\\ \left\langle\begin{array}{l}\texttt{tuple}-\langle 1,0\rangle,\\ \texttt{tuple}-\langle 0,1\rangle,\\ \texttt{tuple}-\langle 0,0\rangle,\\ \texttt{tuple}-\langle 1,1\rangle\end{array}\right\rangle,2\end{array}\right)$$

The cond_lex_cost constraint holds since VECTOR is assigned to the second item of the collection PREFERENCE_TABLE.

Typical

|TUPLE_OF_VALS| > 1
|VECTOR| > 1
|PREFERENCE_TABLE| > 1

Symmetries

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

Usage We consider an example taken from [437] were 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 $\langle$`Naples`, `spring`$\rangle$, $\langle$`Naples`, `summer`$\rangle$, $\langle$`Helsinki`, `spring`$\rangle$ and $\langle$`Helsinki`, `summer`$\rangle$. For instance we may state within the preference table `PREFERENCE_TABLE` of the `cond_lex_cost` constraint the preference ordering $\langle$`Naples`, `spring`$\rangle \succ \langle$`Helsinki`, `summer`$\rangle \succ \langle$`Helsinki`, `spring`$\rangle \succ \langle$`Naples`, `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 5.202 depicts the automaton associated with `cond_lex_lesseq` constraint. Let $\texttt{VAR}_k$ denote the `var` attribute of the k^{th} item of the `VECTOR` collection. Figure 5.203 depicts the reformulation of the `cond_lex_cost` constraint.

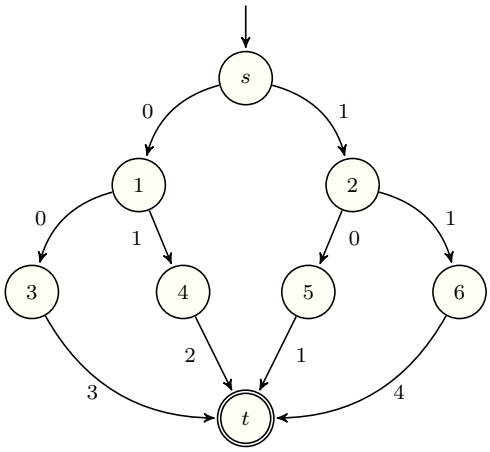

Figure 5.202: Automaton of the `cond_lex_cost` constraint given in the **Example** slot

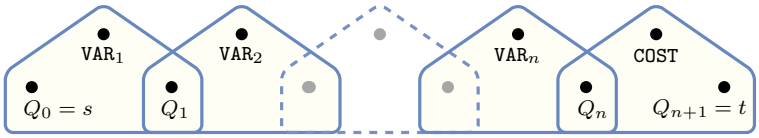

Figure 5.203: Hypergraph of the reformulation corresponding to the automaton of the `cond_lex_cost` constraint