5.67 cond_lex_greater

DESCRIPTION LINKS AUTOMATON

Origin Inspired by [398].

Constraint cond_lex_greater(VECTOR1, VECTOR2, PREFERENCE_TABLE)

Type

```
TUPLE_OF_VALS : collection(val−int)
```

Arguments

```
VECTOR1           : collection(var−dvar)
VECTOR2           : collection(var−dvar)
PREFERENCE_TABLE  : collection(tuple − TUPLE_OF_VALS)
```

Restrictions

```
required(TUPLE_OF_VALS, val)
required(VECTOR1, var)
required(VECTOR2, var)
|VECTOR1| = |VECTOR2|
|VECTOR1| = |TUPLE_OF_VALS|
required(PREFERENCE_TABLE, tuple)
same_size(PREFERENCE_TABLE, tuple)
distinct(PREFERENCE_TABLE, [])
in_relation(VECTOR1, PREFERENCE_TABLE)
in_relation(VECTOR2, PREFERENCE_TABLE)
```

Purpose VECTOR1 and VECTOR2 are both assigned to the $\mathtt{I}^{th}$ and $\mathtt{J}^{th}$ items of the collection PREFERENCE_TABLE such that $\mathtt{I} > \mathtt{J}$.

Example

$$\left\langle \begin{array}{l} \langle 0,0\rangle, \\ \langle 1,0\rangle, \\ \left\langle \begin{array}{l} \mathtt{tuple} - \langle 1,0\rangle, \\ \mathtt{tuple} - \langle 0,1\rangle, \\ \mathtt{tuple} - \langle 0,0\rangle, \\ \mathtt{tuple} - \langle 1,1\rangle \end{array} \right\rangle \end{array} \right\rangle$$

The cond_lex_greater constraint holds since VECTOR1 and VECTOR2 are respectively assigned to the third and first items of the collection PREFERENCE_TABLE.

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, automaton without counters, reified automaton constraint.

constraint network structure: Berge-acyclic constraint network.

constraint type: order constraint.

filtering: arc-consistency.

modelling: preferences.

symmetry: lexicographic order.

Automaton

Figure 5.139 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 5.140 depicts the reformulation of the cond_lex_greater constraint. This reformulation uses:

- Two occurrences of the automaton depicted by Figure 5.139 for computing the positions I and J within the preference table corresponding to VECTOR1 and VECTOR2.
- The binary constraint I $>$ J.

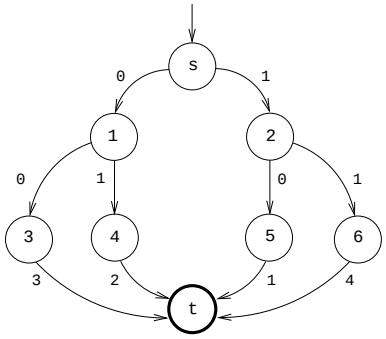

Figure 5.139: Automaton associated with the preference table of the cond_lex_greater constraint given in the example

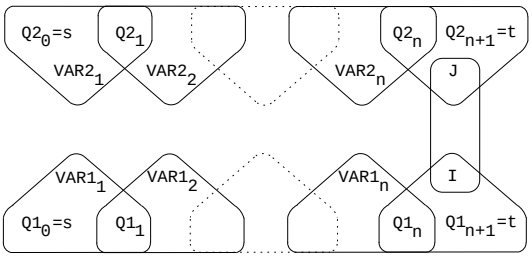

Figure 5.140: Hypergraph of the reformulation corresponding to the cond_lex_greater constraint: it uses two occurrences of the automaton of Figure 5.139 and the constraint I $>$ J