5.228 lex_equal

DESCRIPTION LINKS GRAPH AUTOMATON

Origin Initially introduced for defining `nvector`

Constraint `lex_equal(VECTOR1, VECTOR2)`

Synonyms `equal`, `eq`.

Arguments

`VECTOR1 : collection(var−dvar)`
`VECTOR2 : collection(var−dvar)`

Restrictions

`required(VECTOR1, var)`
`required(VECTOR2, var)`
`|VECTOR1| = |VECTOR2|`

Purpose `VECTOR1` is *equal to* `VECTOR2`. Given two vectors, $\vec{X}$ and $\vec{Y}$ of n components, $\langle X_0, \ldots, X_{n-1}\rangle$ and $\langle Y_0, \ldots, Y_{n-1}\rangle$, $\vec{X}$ is *equal to* $\vec{Y}$ if and only if $n = 0$ or $X_0 = Y_0 \wedge X_1 = Y_1 \wedge \cdots \wedge X_{n-1} = Y_{n-1}$.

Example $(\langle 1, 9, 1, 5\rangle, \langle 1, 9, 1, 5\rangle)$

The `lex_equal` constraint holds since (1) the first component of the first vector is equal to the first component of the second vector, (2) the second component of the first vector is equal to the second component of the second vector, (3) the third component of the first vector is equal to the third component of the second vector and (4) the fourth component of the first vector is equal to the fourth component of the second vector.

Typical

`|VECTOR1|` > 1
`range(VECTOR1.var)` > 1
`range(VECTOR2.var)` > 1

Symmetries

- Arguments are permutable w.r.t. permutation `(VECTOR1, VECTOR2)`.
- Items of `VECTOR1` and `VECTOR2` are permutable (*same permutation used*).

Arg. properties Contractible wrt. `VECTOR1` and `VECTOR2` (*remove items from same position*).

Used in `atleast_nvector`, `atmost_nvector`, `nvector`, `nvectors`.

See also

common keyword: `nvector` (*vector*).

implied by: `vec_eq_tuple`.

implies: `lex_greatereq`, `lex_lesseq`, `same`.

negation: `lex_different`.

specialisation: `vec_eq_tuple` (`variable` *replaced by* `integer` *in second argument*).

Keywords

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

constraint network structure: Berge-acyclic constraint network.

filtering: arc-consistency.

final graph structure: acyclic, bipartite, no loop.

Arc input(s)	VECTOR1 VECTOR2
Arc generator	$PRODUCT(=) \mapsto$ collection(vector1, vector2)
Arc arity	2
Arc constraint(s)	vector1.var = vector2.var
Graph property(ies)	**NARC**= |VECTOR1|
Graph class	• ACYCLIC • BIPARTITE • NO_LOOP

Graph model Parts (A) and (B) of Figure 5.485 respectively show the initial and final graphs associated with the **Example** slot. Since we use the **NARC** graph property, the arcs of the final graph are stressed in bold.

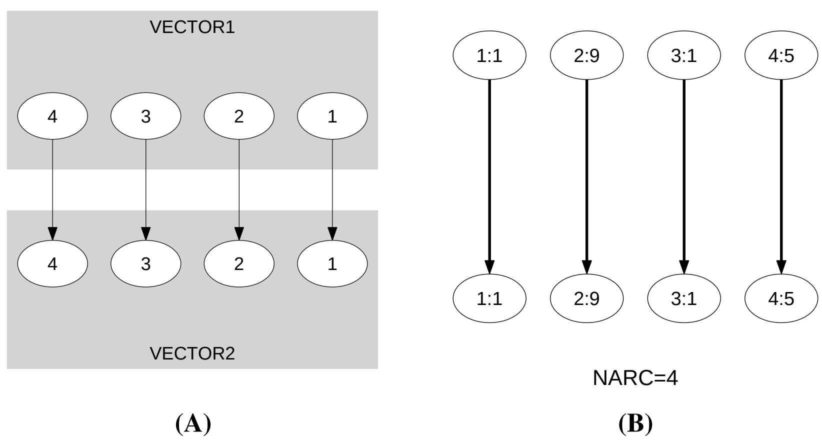

Figure 5.485: Initial and final graph of the lex_equal constraint

Automaton Figure 5.486 depicts the automaton associated with the `lex_equal` constraint. Let $\texttt{VAR1}_i$ and $\texttt{VAR2}_i$ respectively be the `var` attributes of the i^{th} items of the `VECTOR1` and the `VECTOR2` collections. To each pair $(\texttt{VAR1}_i, \texttt{VAR2}_i)$ corresponds a signature variable S_i as well as the following signature constraint: $(\texttt{VAR1}_i \neq \texttt{VAR2}_i \Leftrightarrow S_i = 0) \wedge (\texttt{VAR1}_i = \texttt{VAR2}_i \Leftrightarrow S_i = 1)$.

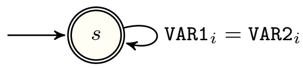

Figure 5.486: Automaton of the `lex_equal` constraint

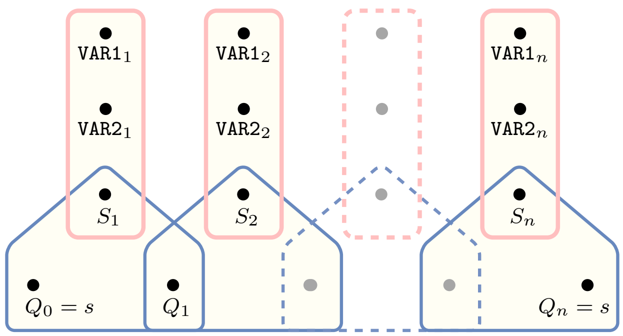

Figure 5.487: Hypergraph of the reformulation corresponding to the automaton of the `lex_equal` constraint