

5.229 lex_equal

	DESCRIPTION	LINKS	GRAPH	AUTOMATON
Origin	Initially introduced for defining <code>nvector</code>			
Constraint	<code>lex_equal(VECTOR1, VECTOR2)</code>			
Synonyms	equal, eq.			
Arguments	VECTOR1 : <code>collection</code> (var-dvar) VECTOR2 : <code>collection</code> (var-dvar)			
Restrictions	<code>required</code> (VECTOR1, var) <code>required</code> (VECTOR2, var) <code> VECTOR1 = VECTOR2 </code>			
Purpose	VECTOR1 is <i>equal</i> to VECTOR2. Given two vectors, \vec{X} and \vec{Y} of n components, $\langle X_0, \dots, X_{n-1} \rangle$ and $\langle Y_0, \dots, Y_{n-1} \rangle$, \vec{X} is <i>equal</i> to \vec{Y} if and only if $n = 0$ or $X_0 = Y_0 \wedge X_1 = Y_1 \wedge \dots \wedge X_{n-1} = Y_{n-1}$.			
Example	$((\langle 1, 9, 1, 5 \rangle, \langle 1, 9, 1, 5 \rangle))$ The <code>lex_equal</code> 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	<code> VECTOR1 > 1</code> <code>range</code> (VECTOR1.var) > 1 <code>range</code> (VECTOR2.var) > 1			
Symmetries	<ul style="list-style-type: none">Arguments are <code>permutable</code> w.r.t. permutation (VECTOR1, VECTOR2).Items of VECTOR1 and VECTOR2 are <code>permutable</code> (<i>same permutation used</i>).			
Arg. properties	<code>Contractible</code> wrt. VECTOR1 and VECTOR2 (<i>remove items from same position</i>).			
Used in	<code>atleast_nvector</code> , <code>atmost_nvector</code> , <code>nvector</code> , <code>nvectors</code> .			
See also	common keyword: <code>nvector</code> (<i>vector</i>). implied by: <code>vec_eq_tuple</code> . implies: <code>lex_greatereq</code> , <code>lex_lesseq</code> , <code>same</code> . negation: <code>lex_different</code> . specialisation: <code>vec_eq_tuple</code> (<i>variable replaced by integer in second argument</i>).			

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	<i>PRODUCT</i> (=) \mapsto collection(vector1,vector2)
Arc arity	2
Arc constraint(s)	vector1.var = vector2.var
Graph property(ies)	NARC = VECTOR1
Graph class	<ul style="list-style-type: none">• ACYCLIC• BIPARTITE• NO_LOOP

Graph model Parts (A) and (B) of Figure 5.504 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.

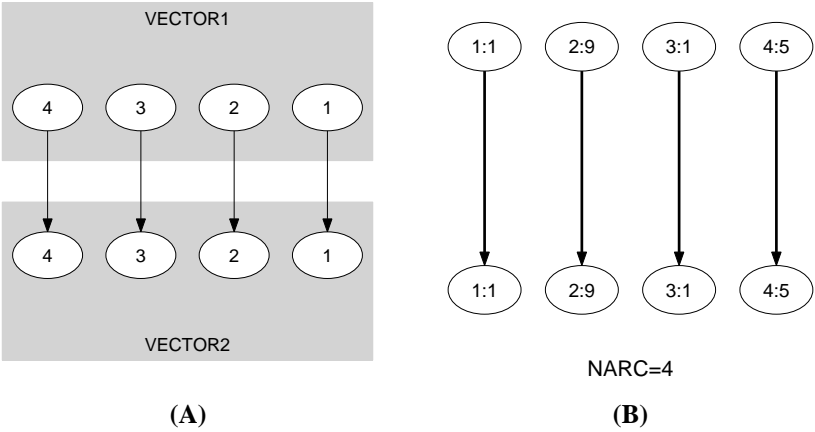
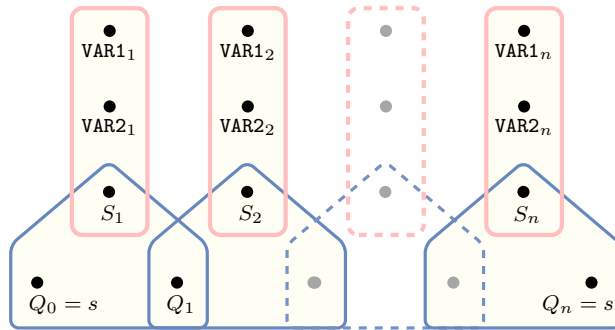


Figure 5.504: Initial and final graph of the `lex_equal` constraint

Automaton

Figure 5.505 depicts the automaton associated with the `lex_equal` constraint. Let VAR1_i and VAR2_i respectively be the `var` attributes of the i^{th} items of the `VECTOR1` and the `VECTOR2` collections. To each pair $(\text{VAR1}_i, \text{VAR2}_i)$ corresponds a signature variable S_i as well as the following signature constraint: $(\text{VAR1}_i \neq \text{VAR2}_i \Leftrightarrow S_i = 0) \wedge (\text{VAR1}_i = \text{VAR2}_i \Leftrightarrow S_i = 1)$.

Figure 5.505: Automaton of the `lex_equal` constraintFigure 5.506: Hypergraph of the reformulation corresponding to the automaton of the `lex_equal` constraint