5.282 not_all_equal

DESCRIPTION LINKS GRAPH AUTOMATON

Origin CHIP

Constraint `not_all_equal(VARIABLES)`

Argument `VARIABLES : collection(var−dvar)`

Restrictions

`required(VARIABLES, var)`
$|$`VARIABLES`$| > 1$

Purpose The variables of the collection `VARIABLES` should take more than a single value.

Example $(\langle 3, 1, 3, 3, 3\rangle)$

The `not_all_equal` constraint holds since the collection $\langle 3, 1, 3, 3, 3\rangle$ involves more than one value (i.e., values 1 and 3).

Typical

$|$`VARIABLES`$| > 2$
`nval(VARIABLES.var)` > 2

Symmetries

- Items of `VARIABLES` are permutable.
- All occurrences of two distinct values of `VARIABLES.var` can be swapped; all occurrences of a value of `VARIABLES.var` can be renamed to any unused value.

Arg. properties Extensible wrt. `VARIABLES`.

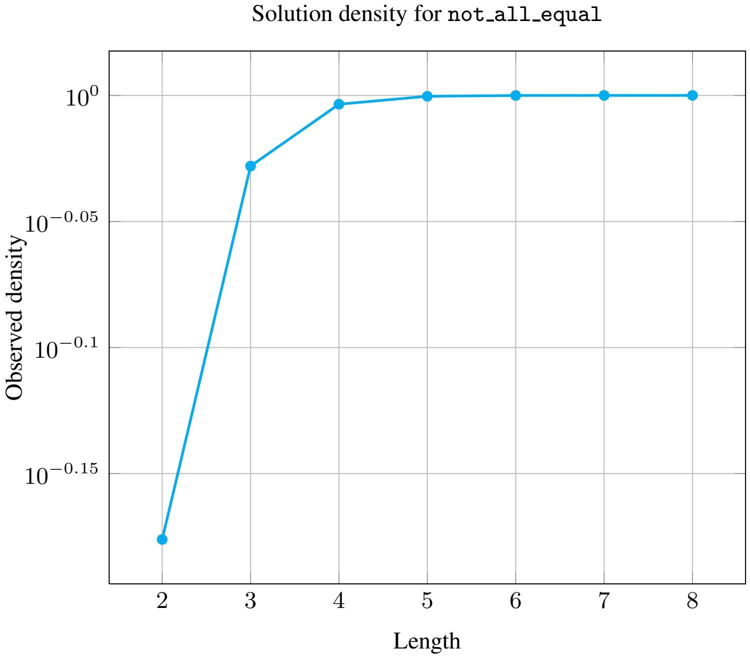
Algorithm If the intersection of the domains of the variables of the `VARIABLES` collection is empty the `not_all_equal` constraint is entailed. Otherwise, when only a single variable V remains not fixed, remove the unique value (unique since the constraint is not entailed) taken by the other variables from the domain of V.

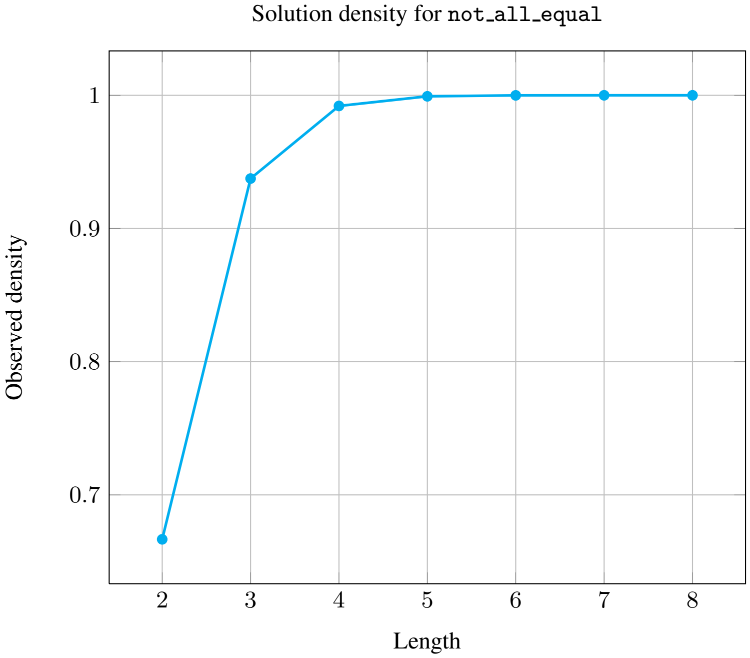
Reformulation The `not_all_equal(VARIABLES)` constraint can be expressed as `atleast_nvalue(2, VARIABLES)`.

Counting

Length (n)	2	3	4	5	6	7	8
Solutions	6	60	620	7770	117642	2097144	43046712

Number of solutions for `not_all_equal`: domains $0..n$





Systems `rel` in **Gecode**.

See also **generalisation:** `nvalue` *(introduce a variable for counting the number of distinct values)*.

implied by: `alldifferent`.

negation: `all_equal`.

specialisation: `neq` *(when go down to two variables)*.

used in reformulation: `atleast_nvalue`.

Keywords

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

constraint network structure: sliding cyclic(1) constraint network(1).

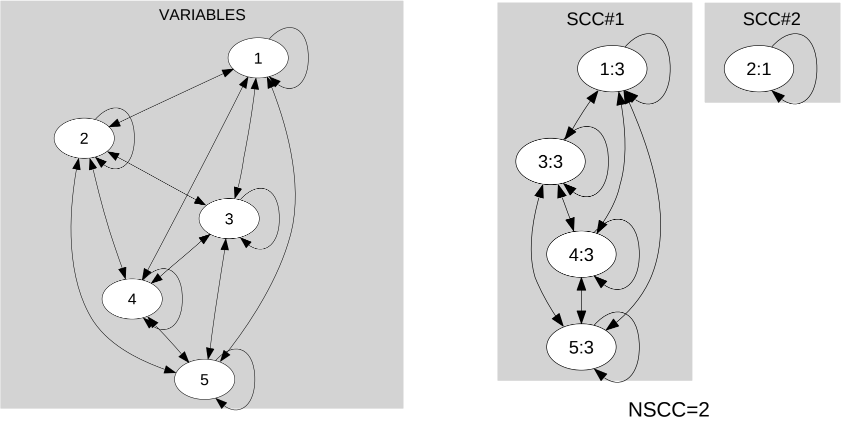
constraint type: value constraint.

filtering: arc-consistency.

final graph structure: equivalence.

Arc input(s)	`VARIABLES`
Arc generator	$CLIQUE \mapsto \texttt{collection}(\texttt{variables1}, \texttt{variables2})$
Arc arity	2
Arc constraint(s)	$\texttt{variables1.var} = \texttt{variables2.var}$
Graph property(ies)	$\mathbf{NSCC} > 1$

Graph model Parts (A) and (B) of Figure 5.589 respectively show the initial and final graph associated with the **Example** slot. Since we use the **NSCC** graph property we show the different strongly connected components of the final graph. Each strongly connected component corresponds to a value that is assigned to some variables of the `VARIABLES` collection. The `not_all_equal` holds since the final graph contains more than one strongly connected component.



(A) **(B)**

Figure 5.589: Initial and final graph of the `not_all_equal` constraint

Automaton Figure 5.590 depicts the automaton associated with the `not_all_equal` constraint. To each pair of consecutive variables $(\texttt{VAR}_i, \texttt{VAR}_{i+1})$ of the collection VARIABLES corresponds a signature variable S_i. The following signature constraint links $\texttt{VAR}_i$, $\texttt{VAR}_{i+1}$ and S_i: $\texttt{VAR}_i = \texttt{VAR}_{i+1} \Leftrightarrow S_i$.

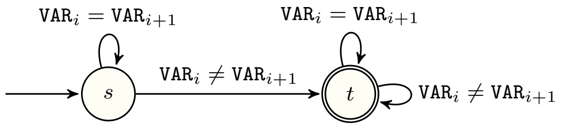

$\texttt{VAR}_i = \texttt{VAR}_{i+1}$
$\texttt{VAR}_i = \texttt{VAR}_{i+1}$
s
$\texttt{VAR}_i \neq \texttt{VAR}_{i+1}$
t
$\texttt{VAR}_i \neq \texttt{VAR}_{i+1}$

Figure 5.590: Automaton of the `not_all_equal` constraint

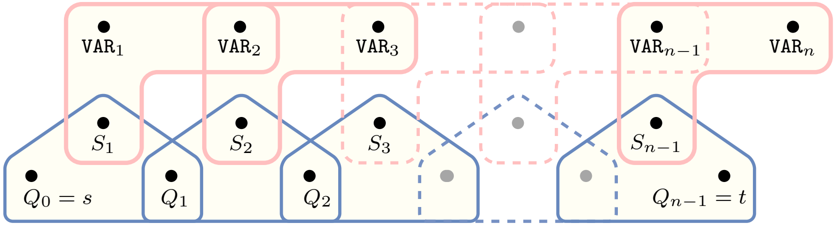


Figure 5.591: Hypergraph of the reformulation corresponding to the automaton of the `not_all_equal` constraint