5.154 equivalent

DESCRIPTION LINKS AUTOMATON

Origin Logic

Constraint equivalent(VAR, VARIABLES)

Synonym eq.

Arguments

VAR	:	dvar
VARIABLES	:	collection(var−dvar)

Restrictions

VAR ≥ 0
VAR ≤ 1
|VARIABLES| $= 2$
required(VARIABLES, var)
VARIABLES.var ≥ 0
VARIABLES.var ≤ 1

Purpose Let VARIABLES be a collection of 0-1 variables $\text{VAR}_1, \text{VAR}_2$. Enforce VAR $= (\text{VAR}_1 \Leftrightarrow \text{VAR}_2)$.

Example

$(1, \langle 0, 0\rangle)$
$(0, \langle 0, 1\rangle)$
$(0, \langle 1, 0\rangle)$
$(1, \langle 1, 1\rangle)$

Symmetries

- Items of VARIABLES are permutable.
- All occurrences of 0 in VAR and in VARIABLES.var can be set to 1.

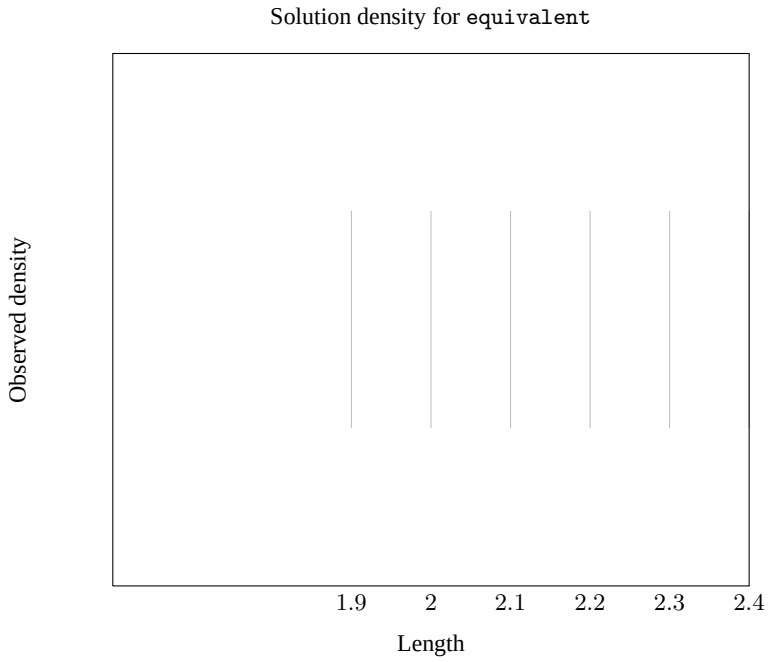
Arg. properties

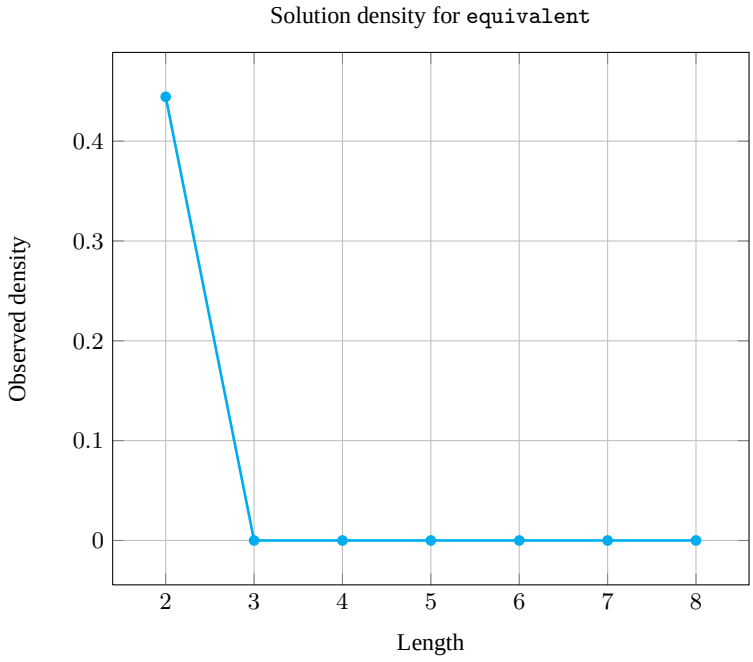
Functional dependency: VAR determined by VARIABLES.

Counting

Length (n)	2	3	4	5	6	7	8
Solutions	4	0	0	0	0	0	0

Number of solutions for equivalent: domains $0..n$

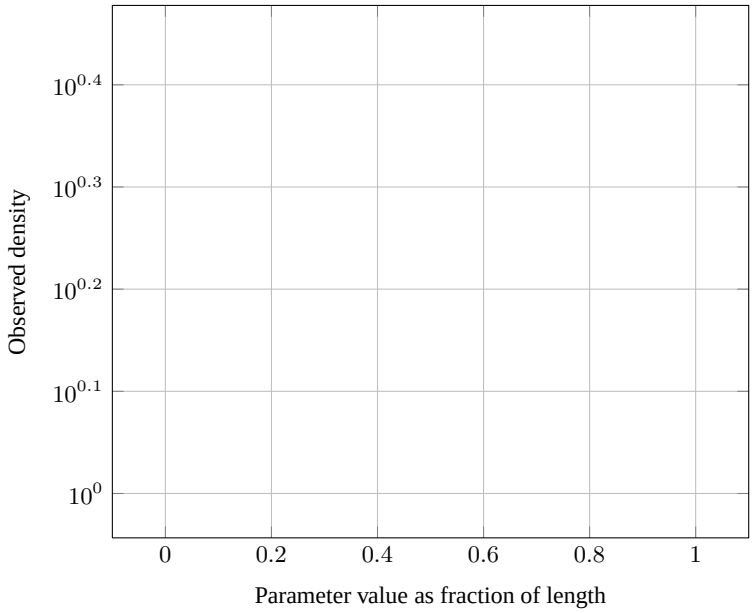

Solution density for equivalent
Observed density
1.9
2
2.1
2.2
2.3
2.4
Length

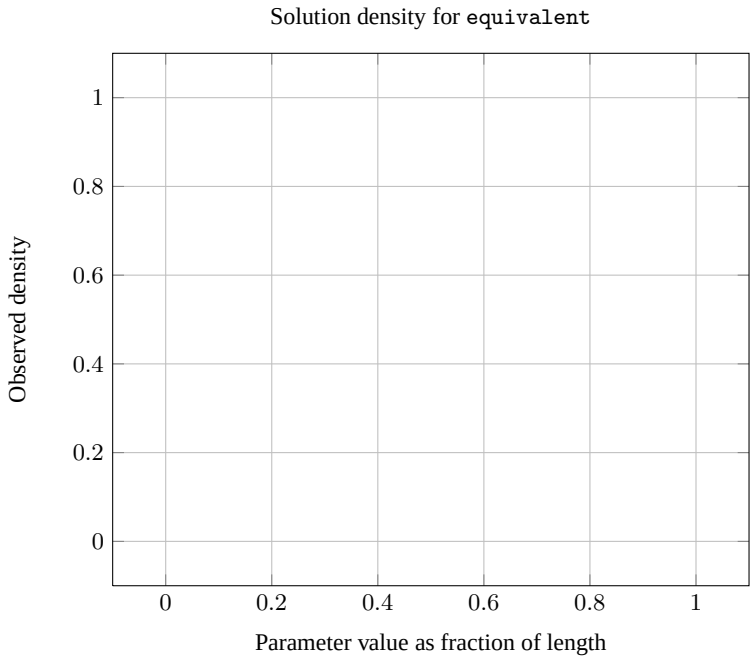

Solution density for equivalent
Observed density
0.4
0.3
0.2
0.1
0
2
3
4
5
6
7
8
Length

Length (n)		2
Total		4
Parameter value	0	2
	1	2

Solution count for `equivalent`: domains $0..n$

Solution density for `equivalent`





Systems `ifOnlyIf` in **Choco**, `rel` in **Gecode**, `eqbool` in **JaCoP**, `#<=>` in **SICStus**.

See also **common keyword:** `and`, `imply`, `nand`, `nor`, `or`, `xor` (*Boolean constraint*).

implies: `atleast_nvalue`, `soft_all_equal_min_ctr`, `soft_alldifferent_ctr`.

Keywords **characteristic of a constraint:** automaton, automaton without counters, reified automaton constraint.

constraint arguments: pure functional dependency.

constraint network structure: Berge-acyclic constraint network.

constraint type: Boolean constraint.

filtering: arc-consistency.

modelling: functional dependency.

Automaton

Figure 5.347 depicts the automaton associated with the `equivalent` constraint. To the first argument `VAR` of the `equivalent` constraint corresponds the first signature variable. To each variable $\texttt{VAR}_i$ of the second argument `VARIABLES` of the `equivalent` constraint corresponds the next signature variable. There is no signature constraint.

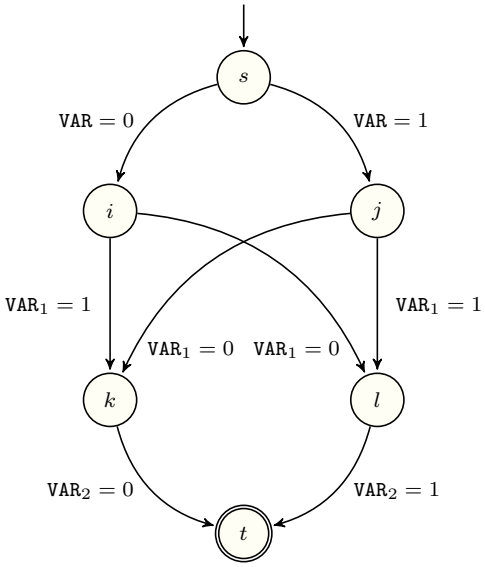


Figure 5.347: Automaton of the `equivalent` constraint

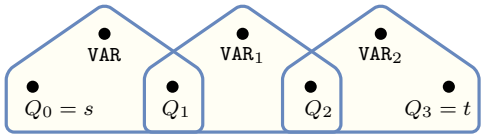


Figure 5.348: Hypergraph of the reformulation corresponding to the automaton of the `equivalent` constraint