5.30 and

DESCRIPTION LINKS AUTOMATON

Origin

Logic

Constraint

and(VAR, VARIABLES)

Synonym

rel.

Arguments

VAR : dvar
VARIABLES : collection(var−dvar)

Restrictions

$\mathtt{VAR} \geq 0$
$\mathtt{VAR} \leq 1$
$|\mathtt{VARIABLES}| \geq 2$
required(VARIABLES, var)
$\mathtt{VARIABLES.var} \geq 0$
$\mathtt{VARIABLES.var} \leq 1$

Purpose

Let VARIABLES be a collection of 0-1 variables $\mathtt{VAR}_1, \mathtt{VAR}_2, \ldots, \mathtt{VAR}_n$ $(n \geq 2)$. Enforce $\mathtt{VAR} = \mathtt{VAR}_1 \wedge \mathtt{VAR}_2 \wedge \cdots \wedge \mathtt{VAR}_n$.

Example

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

All solutions

Figure 5.79 gives all solutions to the following non ground instance of the and constraint: $\mathtt{VAR} \in [\mathbf{0}, \mathbf{1}]$, $\mathtt{V}_1 \in [0, 1]$, $\mathtt{V}_2 = 1$, $\mathtt{V}_3 \in [0, 1]$, $\mathtt{V}_4 = 1$, $\mathtt{and}(\mathtt{VAR}, \langle \mathtt{V}_1, \mathtt{V}_2, \mathtt{V}_3, \mathtt{V}_4 \rangle)$.

① $(\mathbf{0}, \langle 0, 1, 0, 1 \rangle)$
② $(\mathbf{0}, \langle 0, 1, 1, 1 \rangle)$
③ $(\mathbf{0}, \langle 1, 1, 0, 1 \rangle)$
④ $(\mathbf{1}, \langle 1, 1, 1, 1 \rangle)$

Figure 5.79: All solutions corresponding to the non ground example of the and constraint of the **All solutions** slot

Symmetry

Items of VARIABLES are permutable.

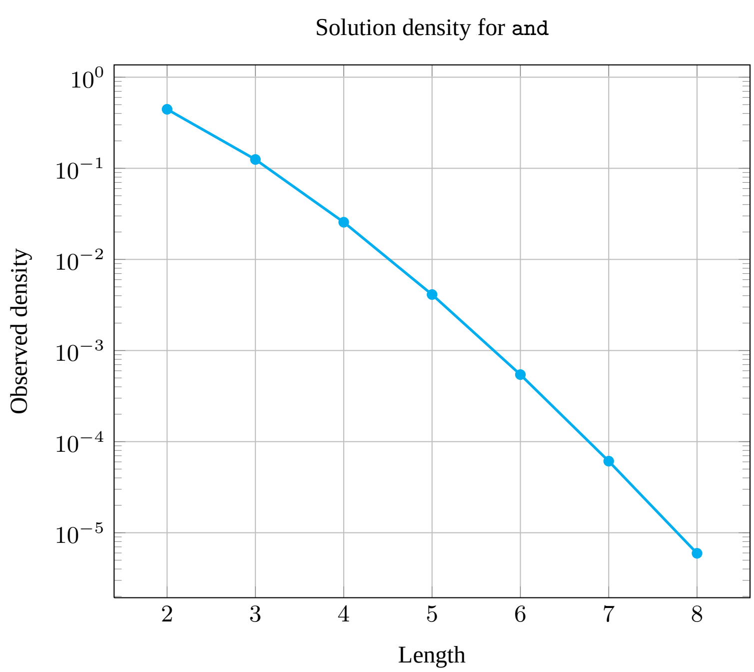
Arg. properties

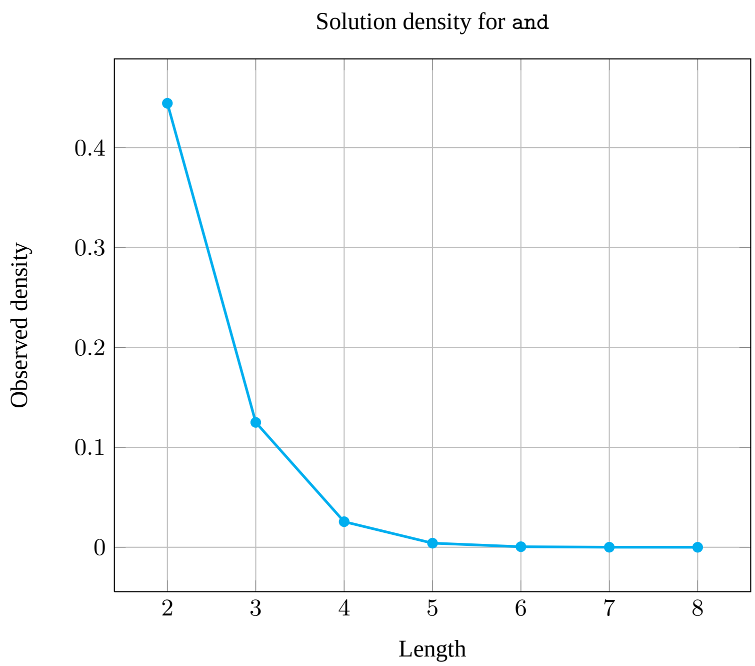
- Functional dependency: `VAR` determined by `VARIABLES`.
- Extensible wrt. `VARIABLES` when `VAR` $= 0$.
- Aggregate: `VAR`($\wedge$), `VARIABLES`(`union`).

Counting

Length (n)	2	3	4	5	6	7	8
Solutions	4	8	16	32	64	128	256

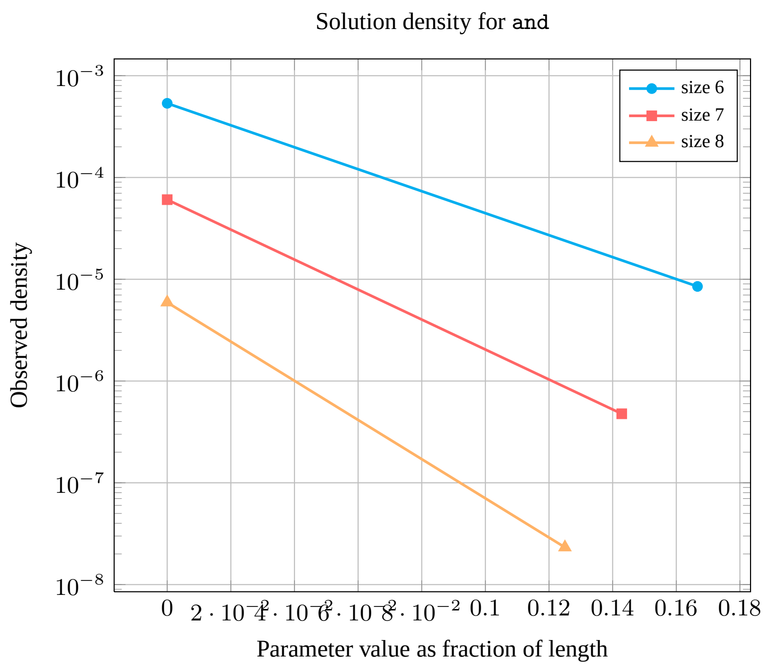
Number of solutions for `and`: domains $0..n$

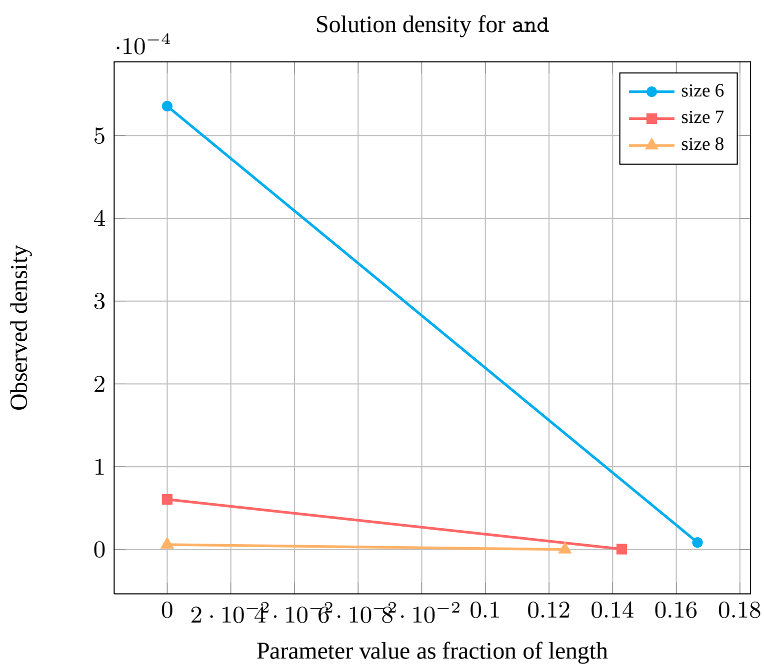




Length (n)		2	3	4	5	6	7	8
Total		4	8	16	32	64	128	256
Parameter value	0	3	7	15	31	63	127	255
	1	1	1	1	1	1	1	1

Solution count for and: domains $0..n$





Systems `reifiedAnd` in **Choco**, `rel` in **Gecode**, `andbool` in **JaCoP**, `#/\` in **SICStus**.

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

implies: atleast_nvalue, between_min_max, minimum, soft_all_equal_min_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.

Cond. implications

- and(VAR, VARIABLES)
  with $|\text{VARIABLES}| > 2$
  implies some_equal(VARIABLES).

- and(VAR, VARIABLES)
  with $\text{VAR} = 0$
  implies nand(VAR, VARIABLES)
  when $\text{VAR} = 1$.

- and(VAR, VARIABLES)
  with $\text{VAR} = 1$
  implies nand(VAR, VARIABLES)
  when $\text{VAR} = 0$.

Automaton

Figure 5.80 depicts a first deterministic automaton without counter associated with the `and` constraint. To the first argument `VAR` of the `and` constraint corresponds the first signature variable. To each variable $\mathtt{VAR}_i$ of the second argument `VARIABLES` of the `and` constraint corresponds the next signature variable. There is no signature constraint.

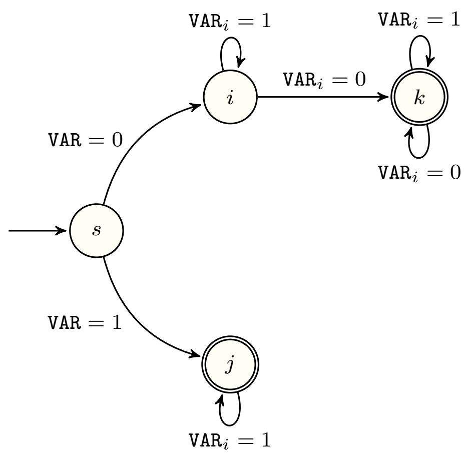


Figure 5.80: Counter free automaton of the $\mathtt{and}(\mathtt{VAR}, \langle \mathtt{VAR}_1, \mathtt{VAR}_2, \ldots, \mathtt{VAR}_n \rangle)$ constraint (the transition $i \xrightarrow{\mathtt{VAR}_i=0} k$ represents the fact that at least one variable $\mathtt{VAR}_i$ should be set to 0 when $\mathtt{VAR} = 0$, while the transition $j \xrightarrow{\mathtt{VAR}_i=1} j$ represents the fact that all $\mathtt{VAR}_i$ should be set to 1 when $\mathtt{VAR} = 1$)

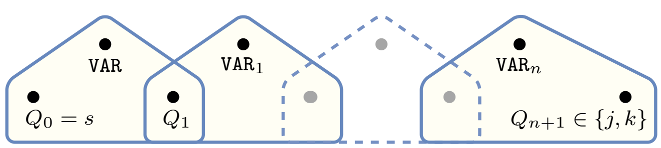


Figure 5.81: Hypergraph of the reformulation corresponding to the automaton of the `and` constraint

Figure 5.82 depicts a second deterministic automaton with one counter associated with the `and` constraint, where the argument `VAR` is unified to the final value of the counter.

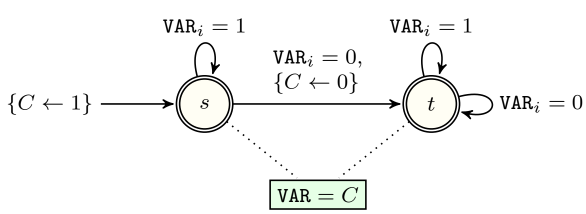


Figure 5.82: Automaton (with one counter) of the `and` constraint

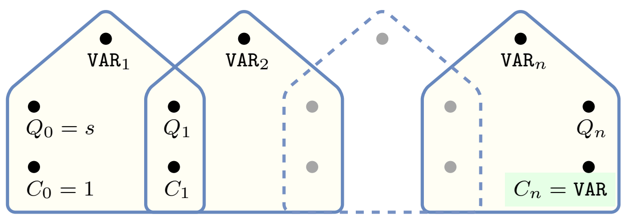


Figure 5.83: Hypergraph of the reformulation corresponding to the automaton (with one counter) of the `and` constraint (since all states of the automaton are accepting there is no restriction on the last variable Q_n)