5.8 all_equal_valley

DESCRIPTION LINKS AUTOMATON

Origin Derived from `valley` and `all_equal`.

Constraint `all_equal_valley(VARIABLES)`

Argument `VARIABLES : collection(var−dvar)`

Restrictions $|\texttt{VARIABLES}| > 0$
`required(VARIABLES, var)`

Purpose A variable V_k $(1 < k < m)$ of the sequence of variables $\texttt{VARIABLES} = V_1, \ldots, V_m$ is a *valley* if and only if there exists an i $(1 < i \leq k)$ such that $V_{i-1} > V_i$ and $V_i = V_{i+1} = \cdots = V_k$ and $V_k < V_{k+1}$.
Enforce all the valleys of the sequence `VARIABLES` to be assigned the same value, i.e. to be located at the same altitude.

Example $(\langle 1, 5, 5, 4, 2, 2, 6, 2, 7\rangle)$

The `all_equal_valley` constraint holds since the two valleys, in bold, of the sequence 1 5 5 4 2 2 6 2 7 are located at the same altitude 2. Figure 5.14 depicts the solution associated with the example.

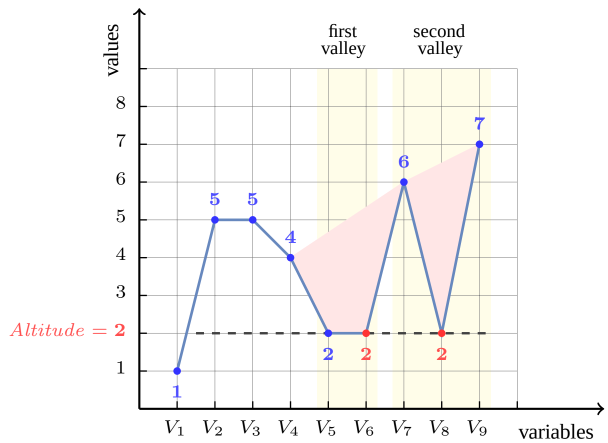


Figure 5.14: Illustration of the **Example** slot: a sequence of nine variables V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_9 respectively fixed to values 1, 5, 5, 4, 2, 2, 6, 2, 7 and its corresponding two valleys, in red, both located at altitude 2

Note that the `all_equal_valley` constraint does not enforce that the minimum value of the sequence `VARIABLES` corresponds to the altitude of its valleys since, as shown by the example, the sequence can starts with an increasing subsequence that start below the altitude of its valleys. It also does not enforce that the sequence VARIABLES contains at least one valley.

All solutions

Figure 5.15 gives all solutions to the following non ground instance of the all_equal_valley constraint: $V_1 \in \{0,5\}$, $V_2 \in [2,3]$, $V_3 = 4$, $V_4 \in [1,2]$, $V_5 \in [4,5]$, all_equal_valley($\langle V_1, V_2, V_3, V_4, V_5 \rangle$).

① ($\langle 0, 2, 4, \mathbf{1}, 4 \rangle$)
② ($\langle 0, 2, 4, \mathbf{1}, 5 \rangle$)
③ ($\langle 0, 2, 4, \mathbf{2}, 4 \rangle$)
④ ($\langle 0, 2, 4, \mathbf{2}, 5 \rangle$)
⑤ ($\langle 0, 3, 4, \mathbf{1}, 4 \rangle$)

⑥ ($\langle 0, 3, 4, \mathbf{1}, 5 \rangle$)
⑦ ($\langle 0, 3, 4, \mathbf{2}, 4 \rangle$)
⑧ ($\langle 0, 3, 4, \mathbf{2}, 5 \rangle$)
⑨ ($\langle 5, \mathbf{2}, 4, \mathbf{2}, 4 \rangle$)
⑩ ($\langle 5, \mathbf{2}, 4, \mathbf{2}, 5 \rangle$)

Figure 5.15: All solutions corresponding to the non ground example of the all_equal_valley constraint of the **All solutions** slot where each valley is coloured in orange

Typical

$|\text{VARIABLES}| \geq 5$
range(VARIABLES.var) > 1
valley(VARIABLES.var) ≥ 2

Symmetries

- Items of VARIABLES can be reversed.
- One and the same constant can be added to the var attribute of all items of VARIABLES.

Arg. properties

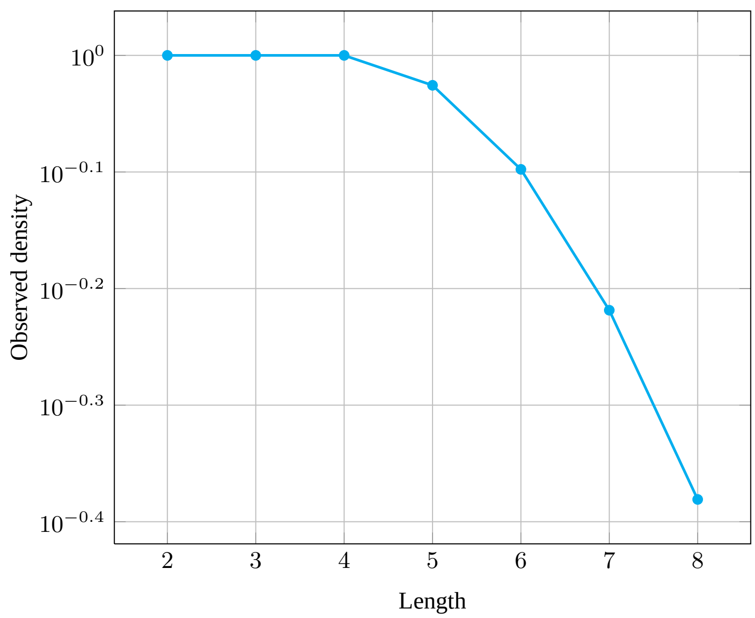
- Prefix-contractible wrt. VARIABLES.
- Suffix-contractible wrt. VARIABLES.

Counting

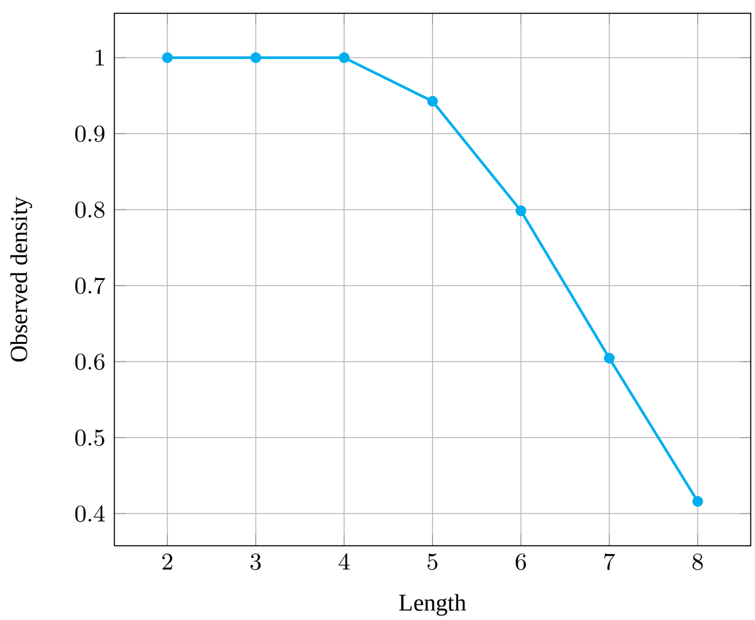
Length (n)	2	3	4	5	6	7	8
Solutions	9	64	625	7330	93947	1267790	17908059

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

Solution density for `all_equal_valley`



Solution density for `all_equal_valley`



See also **implied by:** `all_equal_valley_min`.

implies: `decreasing_valley`, `increasing_valley`.

related: `all_equal_peak`, `valley`.

Keywords **characteristic of a constraint:** automaton, automaton with counters, automaton with same input symbol.

combinatorial object: sequence.

constraint network structure: sliding cyclic(1) constraint network(2).

Cond. implications

- `all_equal_valley`(`VARIABLES`)
  with `valley`(`VARIABLES.var`) > 1
  implies `some_equal`(`VARIABLES`).

- `all_equal_valley`(`VARIABLES`)
  with `valley`(`VARIABLES.var`) > 0
  implies `not_all_equal`(`VARIABLES`).

Automaton Figure 5.16 depicts the automaton associated with the `all_equal_valley` constraint. To each pair of consecutive variables $(\mathtt{VAR}_i, \mathtt{VAR}_{i+1})$ of the collection `VARIABLES` corresponds a signature variable S_i. The following signature constraint links $\mathtt{VAR}_i$, $\mathtt{VAR}_{i+1}$ and S_i: $(\mathtt{VAR}_i < \mathtt{VAR}_{i+1} \Leftrightarrow S_i = 0) \wedge (\mathtt{VAR}_i = \mathtt{VAR}_{i+1} \Leftrightarrow S_i = 1) \wedge (\mathtt{VAR}_i > \mathtt{VAR}_{i+1} \Leftrightarrow S_i = 2)$.

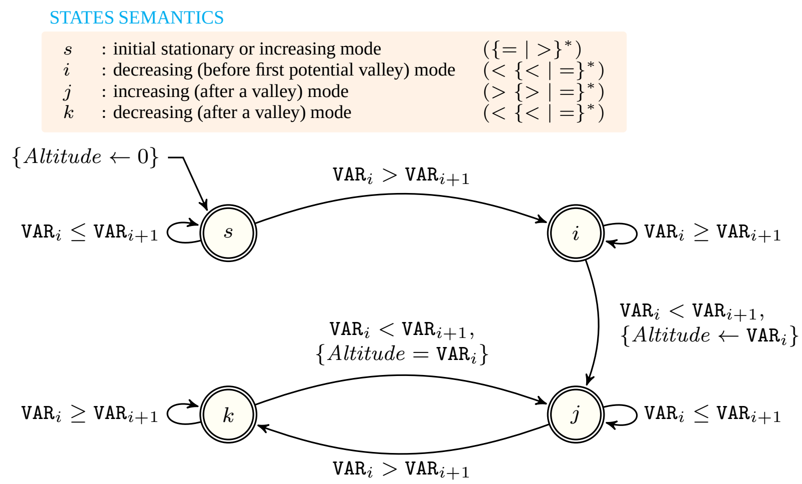


Figure 5.16: Automaton for the `all_equal_valley` constraint (note the conditional transition from state k to state j testing that the counter $Altitude$ is equal to $\mathtt{VAR}_i$ for enforcing that all valleys are located at the same altitude)

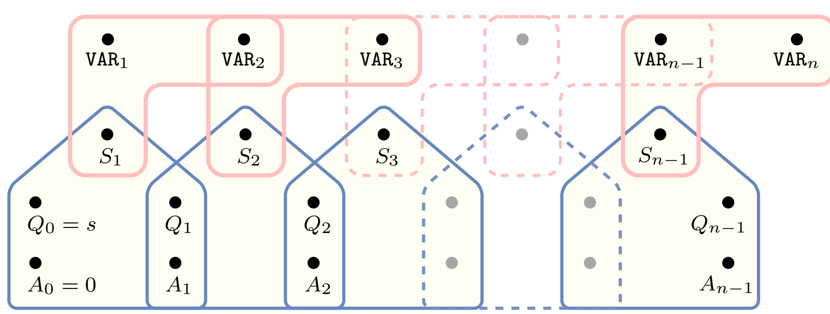


Figure 5.17: Hypergraph of the reformulation corresponding to the automaton of the `all_equal_valley` constraint where A_i stands for the value of the counter $Altitude$ (since all states of the automaton are accepting there is no restriction on the last variable Q_{n-1})