5.349 sliding_card_skip0

DESCRIPTION LINKS GRAPH AUTOMATON

Origin N. Beldiceanu

Constraint `sliding_card_skip0(ATLEAST, ATMOST, VARIABLES, VALUES)`

Arguments

ATLEAST	:	int
ATMOST	:	int
VARIABLES	:	collection(var−dvar)
VALUES	:	collection(val−int)

Restrictions

ATLEAST ≥ 0
ATLEAST $\leq$ |VARIABLES|
ATMOST ≥ 0
ATMOST $\leq$ |VARIABLES|
ATMOST $\geq$ ATLEAST
required(VARIABLES, var)
required(VALUES, val)
distinct(VALUES, val)
VALUES.val $\neq 0$

Purpose

Let n be the total number of variables of the collection VARIABLES. A *maximum non-zero set of consecutive variables* $X_i..X_j (1 \leq i \leq j \leq n)$ is defined in the following way:

- All variables $X_i, \ldots, X_j$ take a non-zero value,
- $i = 1$ or X_{i-1} is equal to 0,
- $j = n$ or X_{j+1} is equal to 0.

Enforces that each maximum non-zero set of consecutive variables of the collection VARIABLES contains at least ATLEAST and at most ATMOST values from the collection of values VALUES.

Example

$(2, 3, \langle 0, 7, 2, 9, 0, 0, 9, 4, 9 \rangle, \langle 7, 9 \rangle)$

The `sliding_card_skip0` constraint holds since the two maximum non-zero set of consecutive values 7 2 9 and 9 4 9 of its third argument $\langle 0, 7, 2, 9, 0, 0, 9, 4, 9 \rangle$ take both 2 ($2 \in$ [ATLEAST, ATMOST] $= [2, 3]$) values within the set of values $\langle 7, 9 \rangle$.

Typical

|VARIABLES| > 1
|VALUES| > 0
|VARIABLES| $>$ |VALUES|
atleast(1, VARIABLES, 0)
ATLEAST $> 0 \vee$ ATMOST $<$ |VARIABLES|

Symmetries

- ATLEAST can be decreased to any value ≥ 0.
- ATMOST can be increased to any value $\leq$ |VARIABLES|.
- Items of VARIABLES can be reversed.
- An occurrence of a value different from 0 of VARIABLES.var that belongs to VALUES.val (resp. does not belong to VALUES.val) can be replaced by any other value different from 0 in VALUES.val (resp. not in VALUES.val).

Usage

This constraint is useful in timetabling problems where the variables are interpreted as the type of job that a person does on consecutive days. Value 0 represents a rest day and one imposes a cardinality constraint on periods that are located between rest periods.

Remark

One cannot initially state a global_cardinality constraint since the rest days are not yet allocated. One can also not use an among_seq constraint since it does not hold for the sequences of consecutive variables that contains at least one rest day.

See also

related: among *(counting constraint on the full sequence)*, global_cardinality *(counting constraint for different values on the full sequence)*.

specialisation: among_low_up *(maximal sequences replaced by the full sequence)*.

Keywords

characteristic of a constraint: automaton, automaton with counters.

combinatorial object: sequence.

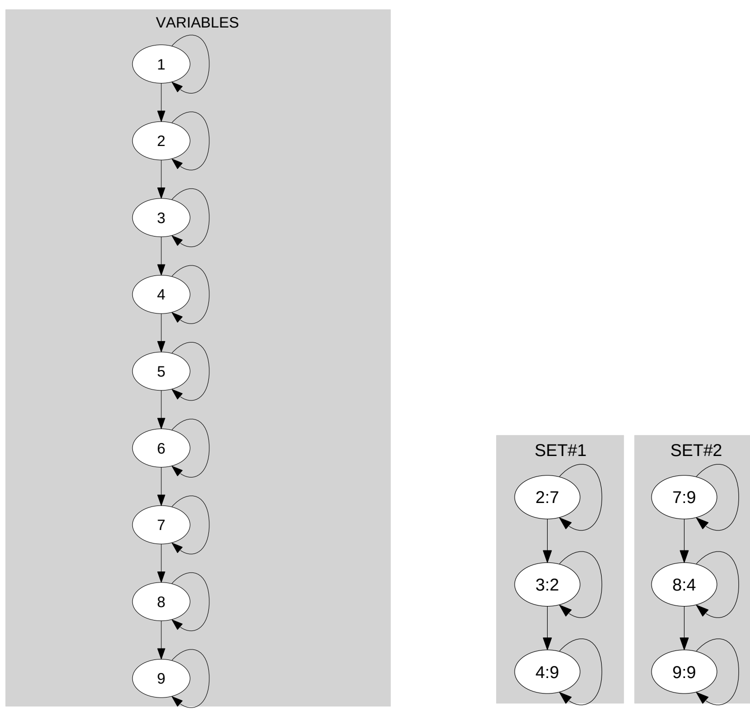
constraint network structure: alpha-acyclic constraint network(2).

constraint type: timetabling constraint, sliding sequence constraint.

Arc input(s)	VARIABLES
Arc generator	$PATH$ ↦ collection(variables1, variables2) $LOOP$ ↦ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	• variables1.var $\neq 0$ • variables2.var $\neq 0$
Sets	CC ↦ [variables]
Constraint(s) on sets	among_low_up(ATLEAST, ATMOST, variables, VALUES)

Graph model Note that the arc constraint will produce the different sequences of consecutive variables that do not contain any 0. The CC set generator produces all the connected components of the final graph.

Parts (A) and (B) of Figure 5.683 respectively show the initial and final graph associated with the **Example** slot. Since we use the set generator CC we show the two connected components of the final graph. Since these two connected components both contains between 2 and 3 variables that take their values in $\{7, 9\}$ the sliding_card_skip0 constraint holds.



(A) **(B)**

Figure 5.683: Initial and final graph of the sliding_card_skip0 constraint

Automaton Figure 5.684 depicts the automaton associated with the `sliding_card_skip0` constraint. To each variable $\texttt{VAR}_i$ of the collection `VARIABLES` corresponds a signature variable S_i. The following signature constraint links $\texttt{VAR}_i$ and S_i:

$(\texttt{VAR}_i = 0) \Leftrightarrow S_i = 0 \wedge$

$(\texttt{VAR}_i \neq 0 \wedge \texttt{VAR}_i \notin \texttt{VALUES}) \Leftrightarrow S_i = 1 \wedge$

$(\texttt{VAR}_i \neq 0 \wedge \texttt{VAR}_i \in \texttt{VALUES}) \Leftrightarrow S_i = 2.$

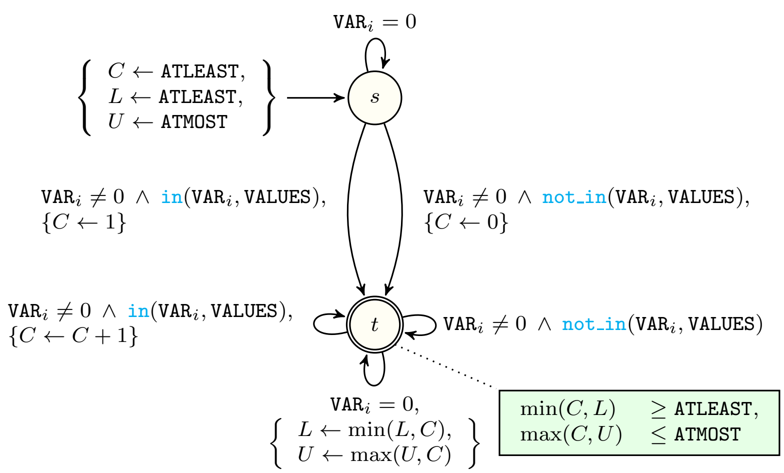


Figure 5.684: Automaton of the `sliding_card_skip0` constraint

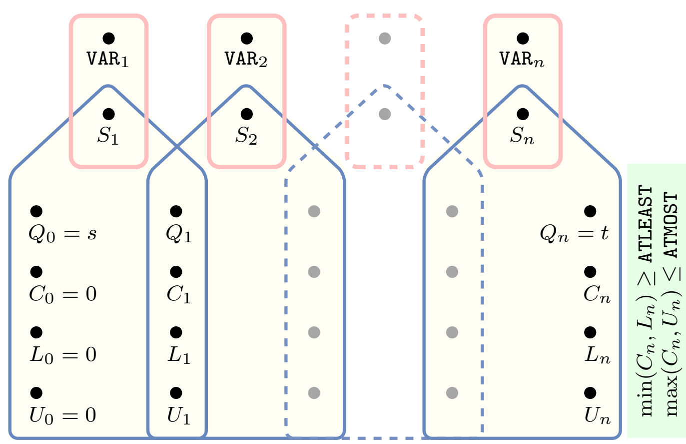


Figure 5.685: Hypergraph of the reformulation corresponding to the automaton of the `sliding_card_skip0` constraint