5.108 cyclic_change_joker

DESCRIPTION LINKS GRAPH AUTOMATON

Origin Derived from cyclic_change.

Constraint cyclic_change_joker(NCHANGE, CYCLE_LENGTH, VARIABLES, CTR)

Arguments

NCHANGE	:	dvar
CYCLE_LENGTH	:	int
VARIABLES	:	collection(var−dvar)
CTR	:	atom

Restrictions

NCHANGE ≥ 0
NCHANGE $<$ |VARIABLES|
CYCLE_LENGTH > 0
required(VARIABLES, var)
VARIABLES.var ≥ 0
CTR $\in [=, \neq, <, \geq, >, \leq]$

Purpose

NCHANGE is the number of times that the following constraint holds:

$$((X+1) \bmod \texttt{CYCLE\_LENGTH})\ \texttt{CTR}\ Y \wedge X < \texttt{CYCLE\_LENGTH} \wedge Y < \texttt{CYCLE\_LENGTH}$$

X and Y correspond to consecutive variables of the collection VARIABLES.

Example

$(2, 4, \langle 3, 0, 2, 4, 4, 4, 3, 1, 4\rangle, \neq)$

Since CTR is set to $\neq$ and since CYCLE_LENGTH is set to 4, a change between two consecutive items X and Y of the VARIABLES collection corresponds to the fact that the condition $((X+1) \bmod 4) \neq Y \wedge X < 4 \wedge Y < 4$ holds. Consequently, the cyclic_change_joker constraint holds since we have the two following changes (i.e., NCHANGE $= 2$) within $\langle 3, 0, 2, 4, 4, 4, 3, 1, 4\rangle$:

- A first change between 0 and 2,
- A second change between 3 and 1.

But when the joker value 4 is involved, there is no change. This is why no change is counted between values 2 and 4, between 4 and 4 and between 1 and 4.

Typical

NCHANGE > 0
CYCLE_LENGTH > 1
|VARIABLES| > 1
range(VARIABLES.var) > 1
maxval(VARIABLES.var) $\geq$ CYCLE_LENGTH
CTR $\in [\neq]$

Symmetry Items of VARIABLES can be shifted.

Arg. properties

Functional dependency: NCHANGE determined by CYCLE_LENGTH, VARIABLES and CTR.

Usage

The cyclic_change_joker constraint can be used in the same context as the cyclic_change constraint with the additional feature: in our example codes 0 to 3 correspond to different type of activities (i.e., working the morning, the afternoon or the night) and code 4 represents a holiday. We want to express the fact that we do not count any change for two consecutive days d_1, d_2 such that d_1 or d_2 is a holiday.

See also

common keyword: change, cyclic_change *(number of changes)*.

implied by: cyclic_change.

Keywords

characteristic of a constraint: cyclic, joker value, automaton, automaton with counters.

constraint arguments: pure functional dependency.

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

constraint type: timetabling constraint.

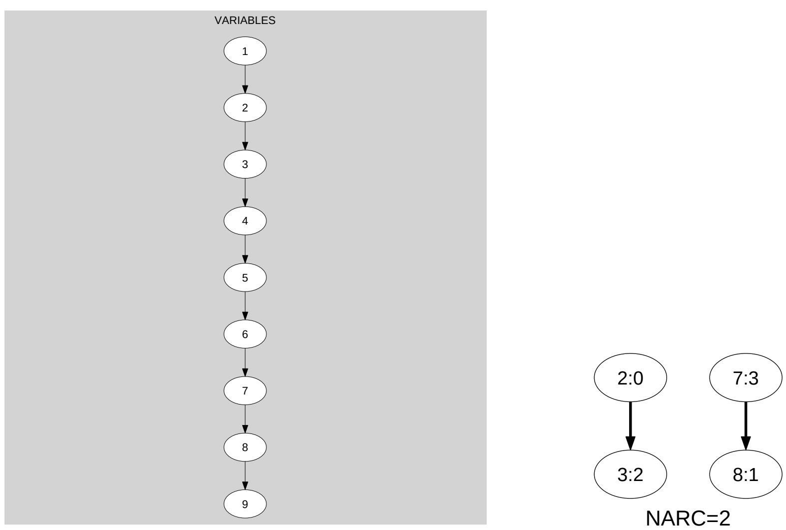
final graph structure: acyclic, bipartite, no loop.

modelling: number of changes, functional dependency.

Arc input(s)	VARIABLES
Arc generator	*PATH* $\mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	• (variables1.var + 1) mod CYCLE_LENGTH CTR variables2.var • variables1.var < CYCLE_LENGTH • variables2.var < CYCLE_LENGTH
Graph property(ies)	**NARC**= NCHANGE
Graph class	• ACYCLIC • BIPARTITE • NO_LOOP

Graph model The *joker values* are those values that are greater than or equal to CYCLE_LENGTH. We do not count any change for those arc constraints involving at least one variable taking a joker value.

Parts (A) and (B) of Figure 5.255 respectively show the initial and final graph associated with the **Example** slot. Since we use the **NARC** graph property, the arcs of the final graph are stressed in bold.



(A) **(B)**

Figure 5.255: Initial and final graph of the cyclic_change_joker constraint

Automaton Figure 5.256 depicts the automaton associated with the `cyclic_change_joker` constraint. To each pair of consecutive variables $(\texttt{VAR}_i, \texttt{VAR}_{i+1})$ of the collection `VARIABLES` corresponds a 0-1 signature variable S_i. The following signature constraint links $\texttt{VAR}_i$, $\texttt{VAR}_{i+1}$ and S_i:

$$(((\texttt{VAR}_i + 1) \bmod \texttt{CYCLE\_LENGTH})\ \texttt{CTR}\ \texttt{VAR}_{i+1} \wedge$$
$$(\texttt{VAR}_i < \texttt{CYCLE\_LENGTH}) \wedge (\texttt{VAR}_{i+1} < \texttt{CYCLE\_LENGTH})) \Leftrightarrow S_i.$$

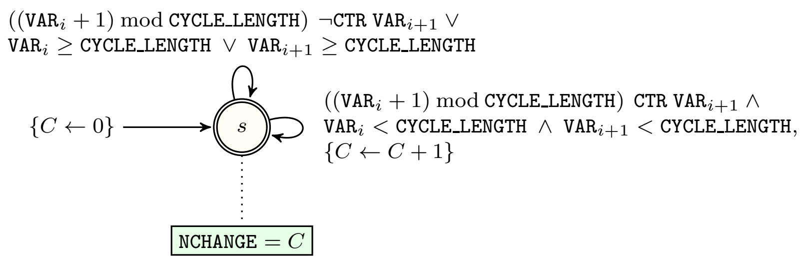


Figure 5.256: Automaton of the `cyclic_change_joker` constraint

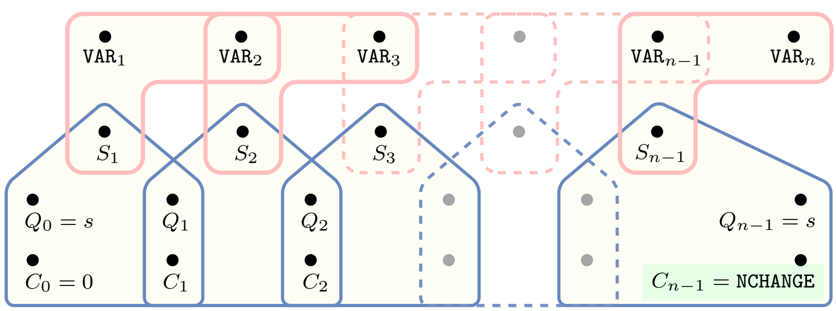


Figure 5.257: Hypergraph of the reformulation corresponding to the automaton of the `cyclic_change_joker` constraint