5.107 cyclic_change

DESCRIPTION LINKS GRAPH AUTOMATON

Origin

Derived from change.

Constraint

cyclic_change(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
VARIABLES.var $<$ CYCLE_LENGTH
CTR $\in [=, \neq, <, \geq, >, \leq]$

Purpose

NCHANGE is the number of times that constraint $((X+1) \bmod$ CYCLE_LENGTH$)$ CTR Y holds; X and Y correspond to consecutive variables of the collection VARIABLES.

Example

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

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 $((X+1) \bmod 4) \neq Y$ holds. Consequently, the cyclic_change constraint holds since we have the two following changes (i.e., NCHANGE $= 2$) within $\langle 3, 0, 2, 3, 1\rangle$:

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

However, the sequence 3 0 does not correspond to a change since $(3+1) \bmod 4$ is equal to 0.

Typical

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

Symmetry

Items of VARIABLES can be shifted.

Arg. properties

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

Usage

This constraint may be used for personnel cyclic timetabling problems where each person has to work according to cycles. In this context each variable of the `VARIABLES` collection corresponds to the type of work a person performs on a specific day. Because of some perturbation (e.g., illness, unavailability, variation of the workload) it is in practice not reasonable to ask for perfect cyclic solutions. One alternative is to use the `cyclic_change` constraint and to ask for solutions where one tries to minimise the number of cycle breaks (i.e., the variable `NCHANGE`).

See also

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

implies: `cyclic_change_joker`.

Keywords

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

constraint arguments: pure functional dependency.

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

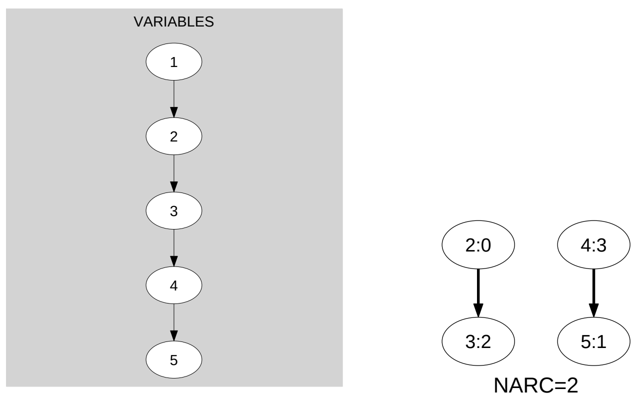
constraint type: timetabling constraint.

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
Graph property(ies)	**NARC**= NCHANGE
Graph class	• ACYCLIC • BIPARTITE • NO_LOOP

Graph model Parts (A) and (B) of Figure 5.252 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 5.252: Initial and final graph of the cyclic_change constraint

Automaton Figure 5.253 depicts the automaton associated with the `cyclic_change` constraint. To each pair of consecutive variables $(\mathtt{VAR}_i, \mathtt{VAR}_{i+1})$ of the collection `VARIABLES` corresponds a 0-1 signature variable S_i. The following signature constraint links $\mathtt{VAR}_i$, $\mathtt{VAR}_{i+1}$ and S_i: $((\mathtt{VAR}_i + 1) \bmod \mathtt{CYCLE\_LENGTH})\ \mathtt{CTR}\ \mathtt{VAR}_{i+1} \Leftrightarrow S_i$.

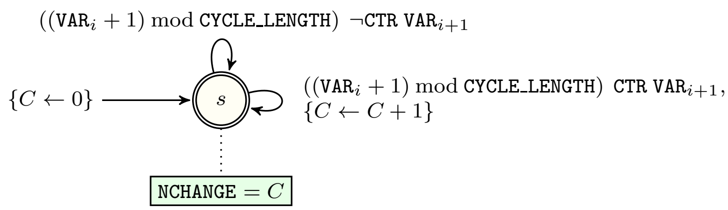


Figure 5.253: Automaton of the `cyclic_change` constraint

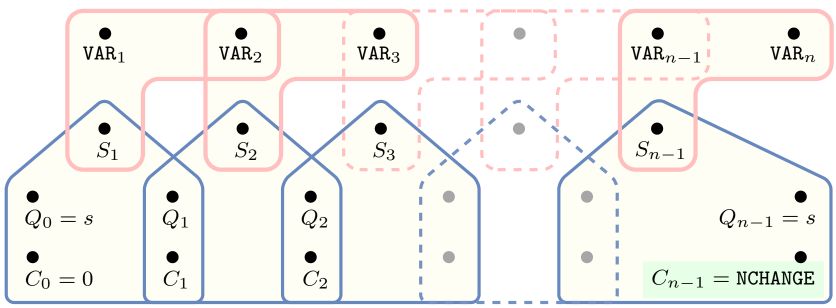


Figure 5.254: Hypergraph of the reformulation corresponding to the automaton of the `cyclic_change` constraint