

5.93 cyclic_change

	DESCRIPTION	LINKS	GRAPH	AUTOMATON
Origin	Derived from change .			
Constraint	<code>cyclic_change(NCHANGE, CYCLE_LENGTH, VARIABLES, CTR)</code>			
Arguments	<pre> NCHANGE : dvar CYCLE_LENGTH : int VARIABLES : collection(var-dvar) CTR : atom </pre>			
Restrictions	<pre> NCHANGE ≥ 0 NCHANGE < VARIABLES CYCLE_LENGTH > 0 required(VARIABLES, var) VARIABLES.var ≥ 0 VARIABLES.var < CYCLE_LENGTH CTR ∈ [=, ≠, <, ≥, >, ≤] </pre>			
Purpose	<div style="border: 1px solid pink; padding: 5px;"> <p>NCHANGE is the number of times that constraint $((X + 1) \bmod \text{CYCLE_LENGTH}) \text{ CTR } Y$ holds; X and Y correspond to consecutive variables of the collection VARIABLES.</p> </div>			
Example	<div style="border: 1px solid blue; padding: 5px; display: inline-block;"> $(2, 4, \langle 3, 0, 2, 3, 1 \rangle, \neq)$ </div> <p>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 <code>cyclic_change</code> constraint holds since we have the two following changes (i.e., NCHANGE = 2) within $\langle 3, 0, 2, 3, 1 \rangle$:</p> <ul style="list-style-type: none"> • A first change between the consecutive values 0 and 2, • A second change between the consecutive values 3 and 1. <p>However, the sequence 3 0 does not correspond to a change since $(3 + 1) \bmod 4$ is equal to 0.</p>			
Typical	<pre> NCHANGE > 0 VARIABLES > 1 range(VARIABLES.var) > 1 </pre>			
Symmetry	Items of VARIABLES can be shifted .			
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*).

Keywords

characteristic of a constraint: [cyclic](#), [automaton](#), [automaton with counters](#).

constraint network structure: [sliding cyclic\(1\)](#) [constraint network\(2\)](#).

constraint type: [timetabling constraint](#).

final graph structure: [acyclic](#), [bipartite](#), [no loop](#).

modelling: [number of changes](#).

Arc input(s)	VARIABLES
Arc generator	<code>PATH</code> \mapsto <code>collection</code> (variables1, variables2)
Arc arity	2
Arc constraint(s)	$(\text{variables1.var} + 1) \bmod \text{CYCLE_LENGTH} \text{ CTR } \text{variables2.var}$
Graph property(ies)	<code>NARC</code> = NCHANGE
Graph class	<ul style="list-style-type: none"> • <code>ACYCLIC</code> • <code>BIPARTITE</code> • <code>NO_LOOP</code>

Graph model

Parts (A) and (B) of Figure 5.185 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.

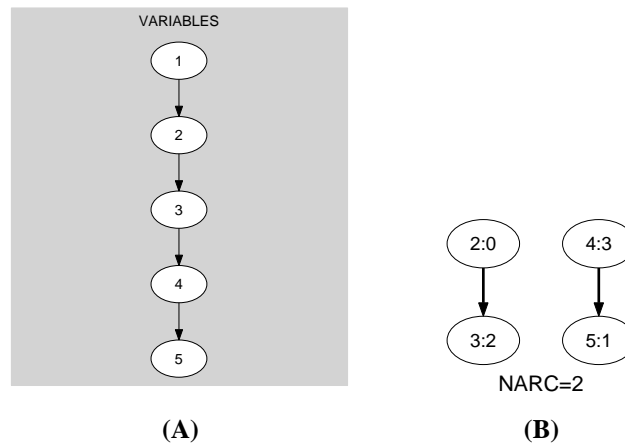


Figure 5.185: Initial and final graph of the `cyclic_change` constraint

Automaton

Figure 5.186 depicts the automaton associated with the `cyclic_change` constraint. To each pair of consecutive variables (VAR_i, VAR_{i+1}) of the collection `VARIABLES` corresponds a 0-1 signature variable S_i . The following signature constraint links VAR_i , VAR_{i+1} and S_i : $((VAR_i + 1) \bmod CYCLE_LENGTH) \text{ CTR } VAR_{i+1} \Leftrightarrow S_i$.

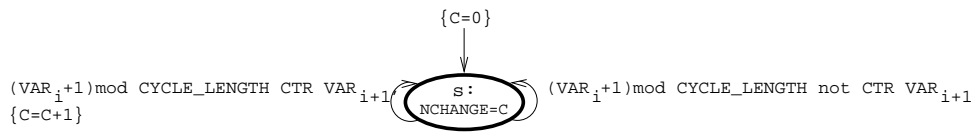


Figure 5.186: Automaton of the `cyclic_change` constraint

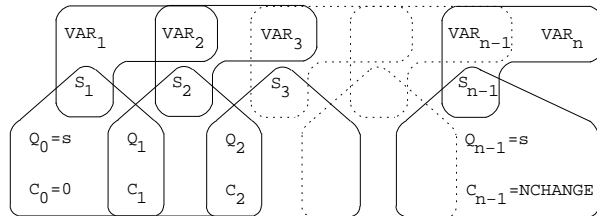


Figure 5.187: Hypergraph of the reformulation corresponding to the automaton of the `cyclic_change` constraint