

5.114 domain_constraint

	DESCRIPTION	LINKS	GRAPH	AUTOMATON
Origin	[306]			
Constraint	domain_constraint(VAR, VALUES)			
Synonym	domain.			
Arguments	VAR : dvar VALUES : collection(var01-dvar, value-int)			
Restrictions	required(VALUES, [var01, value]) VALUES.var01 ≥ 0 VALUES.var01 ≤ 1 distinct(VALUES, value)			
Purpose	Make the link between a domain variable VAR and those 0-1 variables that are associated with each potential value of VAR: The 0-1 variable associated with the value that is taken by variable VAR is equal to 1, while the remaining 0-1 variables are all equal to 0.			
Example	$\left(5, \left\langle \begin{array}{ll} \text{var01} - 0 & \text{value} - 9, \\ \text{var01} - 1 & \text{value} - 5, \\ \text{var01} - 0 & \text{value} - 2, \\ \text{var01} - 0 & \text{value} - 7 \end{array} \right\rangle \right)$			
	The domain_constraint holds since VAR = 5 is set to the value corresponding to the 0-1 variable set to 1, while the other 0-1 variables are all set to 0.			
Typical	VALUES > 1			
Symmetry	Items of VALUES are permutable .			
Usage	This constraint is used in order to make the link between a formulation using finite domain constraints and a formulation exploiting 0-1 variables.			
Reformulation	The domain_constraint(VAR, $\langle \text{var01} - B_1 \text{value} - v_1, \\ \text{var01} - B_2 \text{value} - v_2, \\ \dots \\ \text{var01} - B_{ \text{VALUES} } \text{value} - v_{ \text{VALUES} } \rangle$) constraint can be expressed in term of the following reified constraint (VAR = v ₁ ∧ B ₁ = 1) ∨ (VAR = v ₂ ∧ B ₂ = 1) ∨ ... ∨ (VAR = v _{VALUES} ∧ B _{VALUES} = 1).			
Systems	domainChanneling in Choco , channel in Gecode , in in SICStus , in_set in SICStus .			

See also

common keyword: `link_set_to_booleans` (*channelling constraint*).

related: `roots`.

Keywords

characteristic of a constraint: `automaton`, `automaton without counters`,
`reified automaton constraint`, `derived collection`.

constraint network structure: `centered cyclic(1)` `constraint network(1)`.

constraint type: `decomposition`.

filtering: `linear programming`, `arc-consistency`.

modelling: `channelling constraint`, `domain channel`, `Boolean channel`.

Derived Collection

$$\text{col} \left(\begin{array}{l} \text{VALUE-collection}(\text{var01-int}, \text{value-dvar}), \\ [\text{item}(\text{var01} - 1, \text{value} - \text{VAR})] \end{array} \right)$$
Arc input(s)

VALUE VALUES

Arc generator*PRODUCT* \mapsto collection(value, values)**Arc arity**

2

Arc constraint(s)value.value = values.value \Leftrightarrow values.var01 = 1**Graph property(ies)****NARC** = |VALUES|**Graph model**

The domain_constraint constraint is modelled with the following bipartite graph:

- The first class of vertices corresponds to one single vertex containing the domain variable.
- The second class of vertices contains one vertex for each item of the collection VALUES.

PRODUCT is used in order to generate the arcs of the graph. In our context it takes a collection with one single item (var01 = 1 value = VAR) and the collection VALUES.

The arc constraint between the variable VAR and one potential value v expresses the following:

- If the 0-1 variable associated with v is equal to 1, VAR is equal to v .
- Otherwise, if the 0-1 variable associated with v is equal to 0, VAR is not equal to v .

Since all arc constraints should hold the final graph contains exactly |VALUES| arcs.

Parts (A) and (B) of Figure 5.227 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.

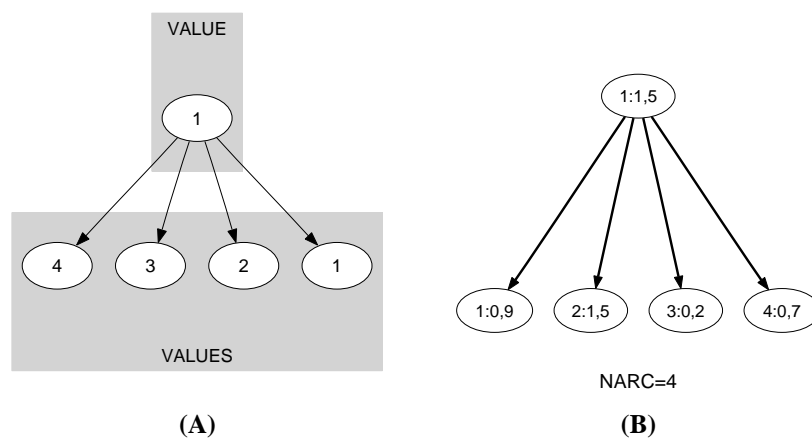


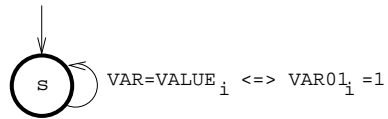
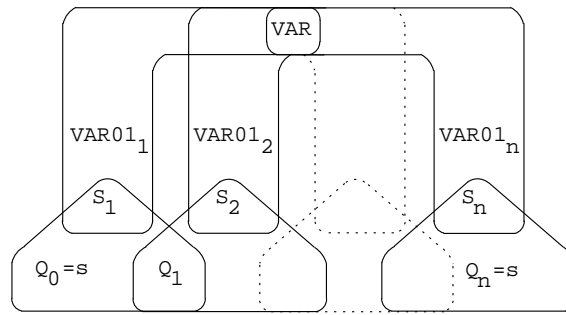
Figure 5.227: Initial and final graph of the domain_constraint constraint

Signature

Since the number of arcs of the initial graph is equal to $VALUES$ the maximum number of arcs of the final graph is also equal to $VALUES$. Therefore we can rewrite the graph property $NARC = |VALUES|$ to $NARC \geq |VALUES|$. This leads to simplify \overline{NARC} to \overline{NARC} .

Automaton

Figure 5.228 depicts the automaton associated with the `domain_constraint` constraint. Let $VAR01_i$ and $VALUE_i$ respectively be the `var01` and the `value` attributes of the i^{th} item of the `VALUES` collection. To each triple $(VAR, VAR01_i, VALUE_i)$ corresponds a 0-1 signature variable S_i as well as the following signature constraint: $((VAR = VALUE_i) \Leftrightarrow VAR01_i) \Leftrightarrow S_i$.

Figure 5.228: Automaton of the `domain_constraint` constraintFigure 5.229: Hypergraph of the reformulation corresponding to the automaton of the `domain_constraint` constraint

20030820

855