

5.139 element

	DESCRIPTION	LINKS	GRAPH	AUTOMATON
Origin	[418]			
Constraint	element(INDEX, TABLE, VALUE)			
Synonyms	nth, element_var, array.			
Arguments	INDEX : dvar TABLE : collection(value—dvar) VALUE : dvar			
Restrictions	INDEX ≥ 1 INDEX ≤ TABLE TABLE > 0 required(TABLE, value)			
Purpose	VALUE is equal to the INDEX th item of TABLE, i.e. VALUE = TABLE[INDEX].			
Example	<div>(3, ⟨6, 9, 2, 9⟩, 2)</div> <p>The <code>element</code> constraint holds since its third argument VALUE = 2 is equal to the 3th (INDEX = 3) item of the collection ⟨6, 9, 2, 9⟩.</p>			
All solutions	<p>Figure 5.318 gives all solutions to the following non ground instance of the <code>element</code> constraint: $I \in [3, 6], V \in [1, 9], \text{element}(I, \langle 4, 8, 1, 0, 3, 3, 4, 3 \rangle, V)$.</p> <div><div>① (3, ⟨4₁, 8₂, 1₃, 0₄, 3₄, 3₆, 7₇, 3₈⟩, 1)</div><div>② (5, ⟨4₁, 8₂, 1₃, 0₄, 3₅, 3₆, 4₇, 3₈⟩, 3)</div><div>③ (6, ⟨4₁, 8₂, 1₃, 0₄, 3₅, 3₆, 4₇, 3₈⟩, 3)</div></div>			
Typical	TABLE > 1 range(TABLE.value) > 1			
Symmetry	All occurrences of two distinct values in TABLE.value or VALUE can be <i>swapped</i> ; all occurrences of a value in TABLE.value or VALUE can be <i>renamed</i> to any unused value.			

Arg. properties

- **Functional dependency**: VALUE determined by INDEX and TABLE.
- **Suffix-extensible** wrt. TABLE.

Usage

See **Usage** slot of `elem`.

Remark

In the original `element` constraint of **CHIP** the `index` attribute was not explicitly present in the table of values. It was implicitly defined as the position of a value in the previous table.

Within some systems (e.g., **Gecode**), the index of the first entry of the table TABLE corresponds to 0 rather than to 1.

When the first entry of the table TABLE corresponds to a value p that is different from 1 we can still use the `element` constraint. We use the reformulation $I = J - p + 1 \wedge \text{element}(I, \text{TABLE}, V)$, where I and J are domain variables respectively ranging from 1 to $|\text{TABLE}|$ and from p to $p + |\text{TABLE}| - 1$.

The `element` constraint is called `nth` in **Choco** (<http://choco.sourceforge.net/>). It is also sometimes called `element_var` when the second argument corresponds to a table of variables.

The `case` constraint [99] is a generalisation of the `element` constraint, where the table is replaced by a directed **acyclic** graph describing the set of solutions: there is a one to one correspondence between the solutions and the paths from the unique source of the dag to its leaves.

Systems

`nth` in **Choco**, `element` in **Gecode**, `element` in **JaCoP**, `element` in **MiniZinc**, `element` in **SICStus**.

See also

common keyword: `elem_from_to`, `element_greatereq`, `element_lesseq`, `element_matrix`, `element_product`, `element_sparse` (*array constraint*), `elementn`, `elements_sparse`, `in_relation`, `stage_element`, `sum` (*data constraint*).

generalisation: `cond_lex_cost` (variable replaced by tuple of variables).

implied by: `elem`.

implies: `elem`.

related: `twin` (*pairs linked by an element with the same table*).

system of constraints: `elements`.

uses in its reformulation: `cycle`, `elements_alldifferent`, `sort_permutation`, `tree_range`, `tree_resource`.

Keywords

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

constraint arguments: pure functional dependency.

constraint network structure: `centered cyclic(2) constraint network(1)`.

constraint type: `data constraint`.

filtering: `arc-consistency`.

heuristics: `labelling by increasing cost`, `regret based heuristics`.

modelling: array constraint, table, functional dependency, variable indexing, variable subscript, disjunction, assignment to the same set of values, sequence dependent set-up.

modelling exercises: assignment to the same set of values, sequence dependent set-up, zebra puzzle.

puzzles: zebra puzzle.

Derived Collection	$\text{col} \left(\begin{array}{l} \text{ITEM} - \text{collection}(\text{index} - \text{dvar}, \text{value} - \text{dvar}), \\ [\text{item}(\text{index} - \text{INDEX}, \text{value} - \text{VALUE})] \end{array} \right)$
Arc input(s)	ITEM TABLE
Arc generator	$\text{PRODUCT} \mapsto \text{collection}(\text{item}, \text{table})$
Arc arity	2
Arc constraint(s)	<ul style="list-style-type: none">• $\text{item.index} = \text{table.key}$• $\text{item.value} = \text{table.value}$
Graph property(ies)	$\text{NARC} = 1$

Graph model The original `element` constraint with three arguments. We use the derived collection `ITEM` for putting together the `INDEX` and `VALUE` parameters of the `element` constraint. Within the arc constraint we use the implicit attribute `key` that associates to each item of a collection its position within the collection.

Parts (A) and (B) of Figure 5.319 respectively show the initial and final graph associated with the **Example** slot. Since we use the **NARC** graph property, the unique arc of the final graph is stressed in bold.

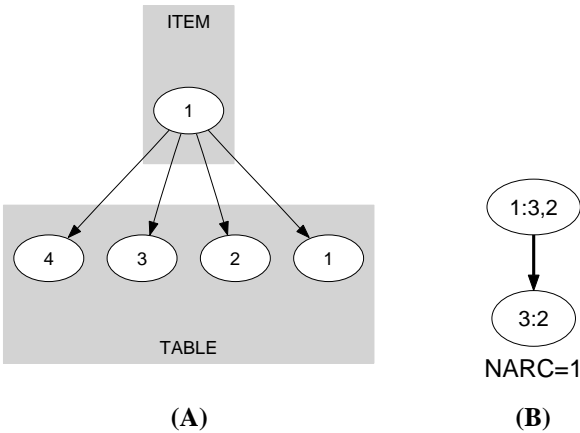


Figure 5.319: Initial and final graph of the `element` constraint

Signature Because of the first condition of the arc constraint the final graph cannot have more than one arc. Therefore we can rewrite $\text{NARC} = 1$ to $\text{NARC} \geq 1$ and simplify NARC to NARC.

Automaton

Figure 5.320 depicts the automaton associated with the `element` constraint. Let $VALUE_i$ be the value attribute of item i of the `TABLE` collection. To each triple $(INDEX, VALUE, VALUE_i)$ corresponds a 0-1 signature variable S_i as well as the following signature constraint: $(INDEX = i \wedge VALUE = VALUE_i) \Leftrightarrow S_i$.

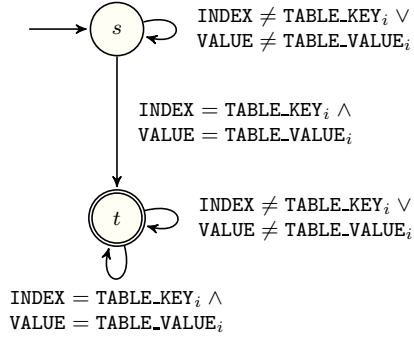


Figure 5.320: Automaton of the `element`(`INDEX`, `TABLE`, `VALUE`) constraint (once one finds the right index and value in the table, one switches from the initial state s to the accepting state t)

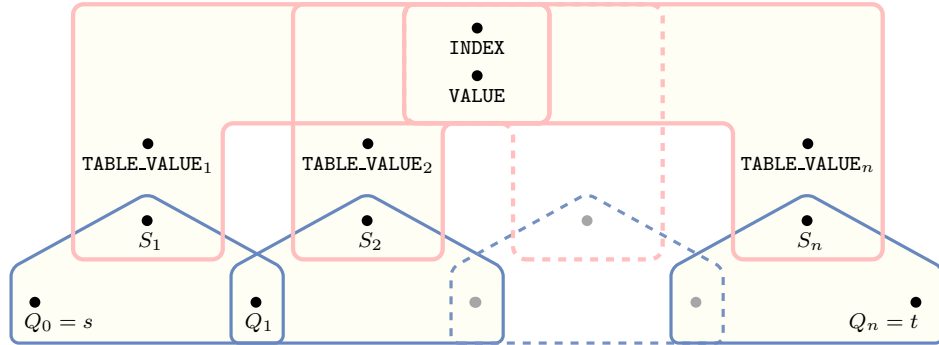


Figure 5.321: Hypergraph of the reformulation corresponding to the automaton of the `element` constraint

Quiz**EXERCISE 1 (checking whether a ground instance holds or not)^a**

- A. Does the constraint `element`(0, $\langle 5, 1, 4, 8, 1 \rangle$, 5) hold?
- B. Does the constraint `element`(3, $\langle 8, 2, 4, 3 \rangle$, 4) hold?
- C. Does the constraint `element`(5, $\langle 0, 1, 2, 3, 4, 5 \rangle$, 5) hold?

^aHint: go back to the definition of `element`.

EXERCISE 2 (finding all solutions)^a

Give all the solutions to the constraint:

$$\begin{cases} I \in [2, 6], \\ V \in [0, 5], \\ \text{element}(I, \langle 0, 2, 9, 5, 9, 2, 3, 9 \rangle, V). \end{cases}$$

^aHint: follow the order induced by the functional dependency between the arguments of `element`, enumerate solutions in lexicographic order.

EXERCISE 3 (finding all solutions)^a

Give all the solutions to the constraint:

$$\begin{cases} I \in [2, 3], \\ V_1 \in [5, 5], V_2 \in [3, 5], V_3 \in [0, 3], \\ V \in [1, 2], \\ \text{element}(I, \langle V_1, V_2, V_3 \rangle, V). \end{cases}$$

^aHint: first find the feasible values of the first argument, then enumerate solutions in lexicographic order.

EXERCISE 4 (identifying infeasible values)^a

Identify all variable-value pairs (V_i, val) ($0 \leq i \leq 3$), such that the following constraint has no solution when value val is assigned to variable V_i :

$$\begin{cases} V_0 \in [2, 3], V_1 \in [2, 4], \\ V_2 \in [0, 4], V_3 \in [3, 5], \\ \text{element}(V_0, \langle V_1, 0, V_2, 6 \rangle, V_3). \end{cases}$$

^aHint: first find the feasible values of the first argument, then filter the other variables.

EXERCISE 5 (variable-based degree of violation)^a

Compute the variable-based degree of violation^b of the following constraints:

- A. `element(0, ⟨2, 2, 2, 2⟩, 2)`,
- B. `element(3, ⟨3, 1, 5, 2, 7⟩, 4)`,
- C. `element(8, ⟨5, 5, 8, 5, 5, 0, 7⟩, 2)`.

^aHint: take advantage of the functional dependency.

^bGiven a constraint for which all variables are fixed, the *variable-based degree of violation* is the minimum number of variables to assign differently in order to satisfy the constraint.

EXERCISE 6 (using entailment for counting)^a

- A. Given an `element`($i, \langle t_1, t_2, \dots, t_n \rangle, v$) constraint where $i, t_1, t_2, \dots, t_n, v$ are variables, what is the minimum number of variables to fix in order to achieve entailment.^b We assume that the constraint has at least one solution.
- B. Exploit entailment in order to compute the number of solutions to the constraint $i \in [1, 3], v_1 \in [0, 1], v_2 \in [1, 9], v_3 \in [3, 5], v \in [2, 7], \text{element}(i, \langle v_1, v_2, v_3 \rangle, v)$.

^aHint: take advantage of the functional dependency, use a case analysis on the first argument.

^bA constraint is *entailed* if and only if it is for sure satisfied even though some of its variables are not fixed.

EXERCISE 7 (modelling with an unconstrained index)^a

What does the `element` constraint model when its first argument, the index, is unconstrained?

^aHint: how would one define the set of solutions of the third argument?

EXERCISE 8 (modelling an index starting at 0)^a

Given a table t whose entries are indexed at $[0, n]$ model the requirement $v = t[i]$.

^aHint: make a shift.

EXERCISE 9 (modelling indirection)^a

Given a table t whose entries vary between 1 and 9, model the requirement $v = t[t[i]]$ as one or several constraints. What is the implicit assumption we have on the entries of the table?

^aHint: use more than one constraint.

SOLUTION TO EXERCISE 1

- A. No, since value the first argument starts at index 1.
- B. Yes, since the third entry of the table is equal to 4.
- C. No, since the fifth entry is equal to 4 (and not to 5).

SOLUTION TO EXERCISE 2

the three solutions

I	$\langle 0, 2, 9, 5, 9, 2, 3, 9 \rangle$	V
①	$\langle 0_1, \mathbf{2}_2, 9_3, 5_4, 9_5, 2_6, 3_7, 9_8 \rangle$	$\mathbf{2}$
②	$\langle 0_1, 2_2, 9_3, \mathbf{5}_4, 9_5, 2_6, 3_7, 9_8 \rangle$	$\mathbf{5}$
③	$\langle 0_1, 2_2, 9_3, 5_4, 9_5, \mathbf{2}_6, 3_7, 9_8 \rangle$	$\mathbf{2}$

1. The active entries of the table are located between index 2 and 6, as shown in bold by $\langle 0_1, \mathbf{2}_2, 9_3, 5_4, 9_5, 2_6, 3_7, 9_8 \rangle$.
2. Among these entries we restrict ourselves to those entries for which the value is located in the domain of variable V , i.e. in interval $[0, 5]$. The remaining entries are shown in bold, i.e. $\langle 0_1, \mathbf{2}_2, 9_3, 5_4, 9_5, 2_6, 3_7, 9_8 \rangle$.
3. This leads to the three solutions $I = \mathbf{2} \ V = \mathbf{2}$, $I = \mathbf{4} \ V = \mathbf{5}$ and $I = \mathbf{6} \ V = \mathbf{2}$.

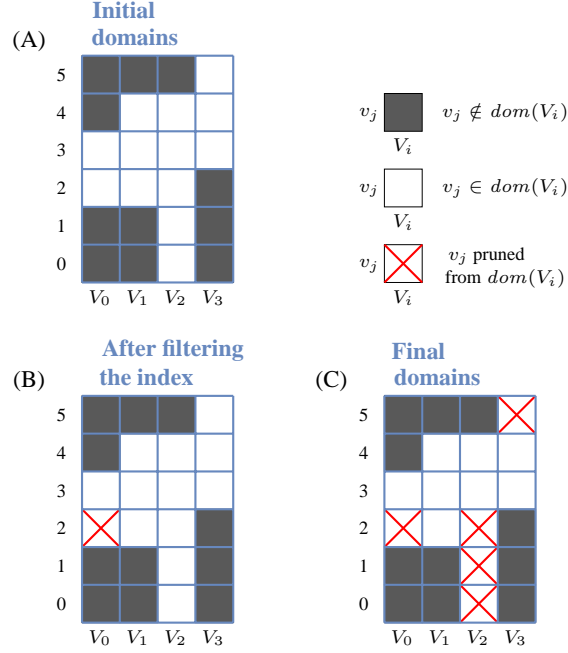
SOLUTION TO EXERCISE 3

the six solutions

I	$\langle V_1, V_2, V_3 \rangle$	V
①	$\langle \mathbf{3}, 5_1, 3_2, \mathbf{1}_3 \rangle$	$\mathbf{1}$
②	$\langle \mathbf{3}, 5_1, 3_2, \mathbf{2}_3 \rangle$	$\mathbf{2}$
③	$\langle \mathbf{3}, 5_1, 4_2, \mathbf{1}_3 \rangle$	$\mathbf{1}$
④	$\langle \mathbf{3}, 5_1, 4_2, \mathbf{2}_3 \rangle$	$\mathbf{2}$
⑤	$\langle \mathbf{3}, 5_1, 5_2, \mathbf{1}_3 \rangle$	$\mathbf{1}$
⑥	$\langle \mathbf{3}, 5_1, 5_2, \mathbf{2}_3 \rangle$	$\mathbf{2}$

1. Since the domain of V_2 which is located at the second entry of the table does not intersect the domain of V (the third argument), the index variable I (the first argument) can not be assigned value 2, and is therefore fixed to 3.
2. Since I is fixed to 3 we have that $V = V_3$. Consequently V and V_3 are assigned a same value that belongs to the intersection of their respective domains, i.e. $[0, 3] \cap [1, 2] = [1, 2]$.

SOLUTION TO EXERCISE 4



1. In part (A) we give the initial domains of the index variable (V_0), of the first and third entries of the table (V_1 , V_2), and of the third argument of the `element` constraint (V_3).
2. In part (B) we prune the index variable V_0 . On the one hand, it can not be assigned value 2 since the second entry of the table is set to 0, and 0 does not belong to the domain of V_3 , see \times . On the other hand, it can be assigned value 3 since $\text{dom}(V_2) \cap \text{dom}(V_3) \neq \emptyset$.
3. Finally in part (C) we remove those values that contradict the fact that $V_2 = V_3$, see \times .

SOLUTION TO EXERCISE 5

- A. The degree of violation is equal to 1 since we only need to change the index from 0 (because 0 is not an allowed value for the index) to any integer value in $[1, 4]$.

$\text{element}^{\overset{1}{\boxed{0}}}(\langle 2, 2, 2, 2 \rangle, 2)$

- B. The degree of violation is equal to 1 since we only need to change the third entry of the table to 4 (or to switch the third argument from 4 to 5).

$\text{element}(3, \langle 3, 1, \overset{4}{\boxed{5}}, 2, 7 \rangle, 4)$

- C. The degree of violation is equal to 2 since we need to change both the index (the table has only 7 entries) and the third argument (value 2 does not occur in the table). Rather than changing the third argument, we may change an entry of the table (e.g., if we set the index to 3 we set the third entry of the table to 2).

$\text{element}^{\overset{1}{\boxed{8}}}(\langle 5, 5, 8, 5, 5, 0, 7 \rangle, \overset{5}{\boxed{2}})$

SOLUTION TO EXERCISE 6

- A. We need to fix 3 variables in the following way:

- (i) The first argument, the index i , is fixed to a value α ($1 \leq \alpha \leq n$) such that $\text{dom}(t_\alpha) \cap \text{dom}(v) \neq \emptyset$.
- (ii) We fix the third argument v to a value β in $\text{dom}(t_\alpha) \cap \text{dom}(v)$.
- (iii) We fix t_α to β .

- B. We have 90 solutions depending on whether $i = 1$, $i = 2$, $i = 3$ (and $v = v_i$):

- (i) $|\text{dom}(v) \cap \text{dom}(v_1)| \cdot |\text{dom}(v_2)| \cdot |\text{dom}(v_3)| = 0 \cdot 9 \cdot 3 = 0$,
- (ii) $|\text{dom}(v) \cap \text{dom}(v_2)| \cdot |\text{dom}(v_1)| \cdot |\text{dom}(v_3)| = 6 \cdot 2 \cdot 3 = 36$,
- (iii) $|\text{dom}(v) \cap \text{dom}(v_3)| \cdot |\text{dom}(v_1)| \cdot |\text{dom}(v_2)| = 3 \cdot 2 \cdot 9 = 54$.

SOLUTION TO EXERCISE 7

Given a table t of n entries $t[1], t[2], \dots, t[n]$, **element** models a disjunction stating that the third argument v is equal to one of the values that can be assigned to one of the variables of the table, i.e. $v = t[1] \vee v = t[2] \vee \dots \vee v = t[n]$.

SOLUTION TO EXERCISE 8

The requirement $v = t[i]$ can be modelled as the conjunction of the two constraints $j = i + 1$, `element`($j, \langle t[0], t[1], \dots, t[n] \rangle, v$).

SOLUTION TO EXERCISE 9

The requirement $v = t[t[i]]$ can be modelled as the conjunction of two `element` constraints sharing the same table, namely:

`element`($i, \langle t[1], t[2], \dots, t[9] \rangle, j$) \leftarrow *inner* indirection $t[t[i]]$
`element`($j, \langle t[1], t[2], \dots, t[9] \rangle, v$) \leftarrow *outer* indirection $t[t[i]]$

The second `element` constraint assumes that j corresponds to a valid index of the table, i.e. a value between 1 and 9.

20000128

1149