5.137 elem

DESCRIPTION LINKS GRAPH AUTOMATON

Origin Derived from element.

Constraint elem(ITEM, TABLE)

Usual name element

Synonyms nth, array.

Arguments

ITEM : collection(index−dvar, value−dvar)
TABLE : collection(index−int, value−dvar)

Restrictions

required(ITEM, [index, value])
$\texttt{ITEM.index} \geq 1$
$\texttt{ITEM.index} \leq |\texttt{TABLE}|$
$|\texttt{ITEM}| = 1$
$|\texttt{TABLE}| > 0$
required(TABLE, [index, value])
$\texttt{TABLE.index} \geq 1$
$\texttt{TABLE.index} \leq |\texttt{TABLE}|$
distinct(TABLE, index)

Purpose ITEM is equal to one of the entries of the table TABLE.

Example

$$\left(\begin{array}{l}\langle \texttt{index}-3\ \texttt{value}-2\rangle, \\ \left\langle\begin{array}{ll}\texttt{index}-1 & \texttt{value}-6, \\ \texttt{index}-2 & \texttt{value}-9, \\ \texttt{index}-3 & \texttt{value}-2, \\ \texttt{index}-4 & \texttt{value}-9\end{array}\right\rangle\end{array}\right)$$

The elem constraint holds since its first argument ITEM corresponds to the third item of the TABLE collection.

Typical

$|\texttt{TABLE}| > 1$
range(TABLE.value) > 1

Symmetries

- Items of TABLE are permutable.
- All occurrences of two distinct values in ITEM.value or TABLE.value can be swapped; all occurrences of a value in ITEM.value or TABLE.value can be renamed to any unused value.

Arg. properties Functional dependency: ITEM.value determined by ITEM.index and TABLE.

Usage Makes the link between the discrete decision variable `INDEX` and the variable `VALUE` according to a given table of values `TABLE`. We now give five typical uses of the `elem` constraint.

1. In some problems we may have to *represent a function* $y = f(x)$ (with $x \in [1, m]$). In this context we generate the following `elem` constraint where `INDEX` is a domain variable taking its values in $\{1, 2, \ldots, m\}$:

$$\texttt{elem}\left(\begin{array}{l}\langle\ \texttt{index}-x\quad \texttt{value}-y\ \rangle,\\ \left\langle\begin{array}{ll}\texttt{index}-1 & \texttt{value}-f(1),\\ \texttt{index}-2 & \texttt{value}-f(2),\\ & \vdots\\ \texttt{index}-m & \texttt{value}-f(m)\end{array}\right\rangle\end{array}\right)$$

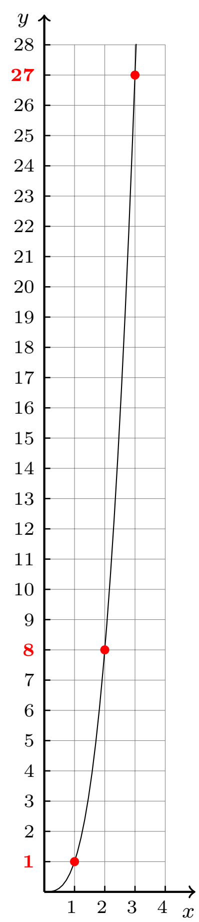


Figure 5.294: $y = x^3$ $(1 \leq x \leq 3)$

As an example, consider the problem of finding the smallest integer that can be decomposed in two different ways in the sum of two cubes [202]. The `elem` constraint can be used for representing the function $y = x^3$ (Figure 5.294). The unique solution $1729 = 12^3 + 1^3 = 10^3 + 9^3$ can be obtained by the following set of constraints:

$$\begin{cases} \texttt{elem}(\langle \texttt{index} - x_1\ \texttt{value} - y_1 \rangle, \\ \qquad \langle \texttt{index} - 1\ \texttt{value} - 1, \texttt{index} - 2\ \texttt{value} - 8, \ldots, \texttt{index} - 20\ \texttt{value} - 8000 \rangle) \\ \texttt{elem}(\langle \texttt{index} - x_2\ \texttt{value} - y_2 \rangle, \\ \qquad \langle \texttt{index} - 1\ \texttt{value} - 1, \texttt{index} - 2\ \texttt{value} - 8, \ldots, \texttt{index} - 20\ \texttt{value} - 8000 \rangle) \\ \texttt{elem}(\langle \texttt{index} - x_3\ \texttt{value} - y_3 \rangle, \\ \qquad \langle \texttt{index} - 1\ \texttt{value} - 1, \texttt{index} - 2\ \texttt{value} - 8, \ldots, \texttt{index} - 20\ \texttt{value} - 8000 \rangle) \\ \texttt{elem}(\langle \texttt{index} - x_4\ \texttt{value} - y_4 \rangle, \\ \qquad \langle \texttt{index} - 1\ \texttt{value} - 1, \texttt{index} - 2\ \texttt{value} - 8, \ldots, \texttt{index} - 20\ \texttt{value} - 8000 \rangle) \\ y_1 + y_2 = y_3 + y_4 \\ x_1 < x_2 \\ x_3 < x_4 \\ x_1 < x_3 \end{cases}$$

The last three inequalities constraints in the conjunction are used for breaking symmetries. The constraints $x_1 < x_2$ and $x_3 < x_4$ respectively order the pairs of variables (x_1, x_2) and (x_3, x_4) from which the sums $x_1^3 + x_2^3$ and $x_3^3 + x_4^3$ are generated. Finally the inequality $x_1 < x_3$ enforces a lexicographic ordering between the two pairs of variables (x_1, x_2) and (x_3, x_4).

2. In some optimisation problems a classical use of the `elem` constraint consists *expressing the link between a discrete choice and its corresponding cost*. For each discrete choice we create an `elem` constraint of the form:

$$\texttt{elem}\left(\begin{array}{l} \langle\ \texttt{index} - \texttt{Choice} \quad \texttt{value} - \texttt{Cost}\ \rangle, \\ \left\langle \begin{array}{ll} \texttt{index} - 1 & \texttt{value} - \texttt{Cost}_1, \\ \texttt{index} - 2 & \texttt{value} - \texttt{Cost}_2, \\ & \vdots \\ \texttt{index} - m & \texttt{value} - \texttt{Cost}_m \end{array} \right\rangle \end{array} \right)$$

where:

- `Choice` is a domain variable that indicates which alternative will be finally selected,
- `Cost` is a domain variable that corresponds to the cost of the decision associated with the value of the `Choice` variable,
- $\texttt{Cost}_1, \texttt{Cost}_2, \ldots, \texttt{Cost}_m$ are the respective costs associated with the alternatives $1, 2, \ldots, m$.

3. In some problems we need to express a disjunction of the form $\texttt{VAR} = \texttt{VAR}_1 \vee \texttt{VAR} = \texttt{VAR}_2 \vee \cdots \vee \texttt{VAR} = \texttt{VAR}_n$. This can be directly reformulated as the following `elem` constraint, where `INDEX` is a domain variable taking its value in the finite set $\{1, 2, \ldots, n\}$ and where the `TABLE` argument corresponds to the domain variables $\texttt{VAR}_1, \texttt{VAR}_2, \ldots, \texttt{VAR}_n$:

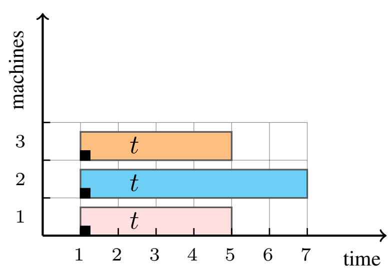
$$\texttt{elem}\left(\begin{array}{l} \langle\ \texttt{index} - \texttt{INDEX} \quad \texttt{value} - \texttt{VAR}\ \rangle, \\ \left\langle \begin{array}{ll} \texttt{index} - 1 & \texttt{value} - \texttt{VAR}_1, \\ \texttt{index} - 2 & \texttt{value} - \texttt{VAR}_2, \\ & \vdots \\ \texttt{index} - n & \texttt{value} - \texttt{VAR}_n \end{array} \right\rangle \end{array} \right)$$

4. In some scheduling problems the duration of a task depends on the machine where the task will be assigned in final schedule. In this case we generate for each task an `elem` constraint of the following form:

$$\texttt{elem}\left(\begin{array}{l}\langle\ \texttt{index}-\texttt{Machine}\quad \texttt{value}-\texttt{Duration}\ \rangle,\\ \left\langle\begin{array}{ll}\texttt{index}-1 & \texttt{value}-\texttt{Dur}_1,\\ \texttt{index}-2 & \texttt{value}-\texttt{Dur}_2,\\ & \vdots\\ \texttt{index}-m & \texttt{value}-\texttt{Dur}_m\end{array}\right\rangle\end{array}\right)$$

where:

- `Machine` is a domain variable that indicates the resource to which the task will be assigned,
- `Duration` is a domain variable that corresponds to the duration of the task,
- $\texttt{Dur}_1, \texttt{Dur}_2, \ldots, \texttt{Dur}_m$ are the respective duration of the task according to the hypothesis that it runs on machine $1, 2$ or m.



$\texttt{Machine} = 3 \Rightarrow \texttt{Duration} = \texttt{Dur}_3 = 4$

$\texttt{Machine} = 2 \Rightarrow \texttt{Duration} = \texttt{Dur}_2 = 6$

$\texttt{Machine} = 1 \Rightarrow \texttt{Duration} = \texttt{Dur}_1 = 4$

$$\texttt{elem}\left(\begin{array}{l}\langle\texttt{index}-\texttt{Machine}\ \texttt{value}-\texttt{Duration}\rangle,\\ \langle\texttt{index}-1\ \texttt{value}-4, \texttt{index}-2\ \texttt{value}-6, \texttt{index}-3\ \texttt{value}-4\rangle\end{array}\right)$$

Figure 5.295: A task t for which the duration depends on the machine 1, 2 or 3 to which it is assigned

Figure 5.295 illustrates this particular use of the `elem` constraint for modelling that a task has a duration of 4, 6 and 4 when we respectively assign it on machines 1, 2 and 3.

5. In some vehicle routing problems we typically use the `elem` constraint to express the distance between location i and the next location visited by a vehicle. For this purpose we generate for each location i an `elem` constraint of the form:

$$\texttt{elem}\left(\begin{array}{l}\langle\ \texttt{index}-\texttt{Next}_i\quad \texttt{value}-\texttt{distance}_i\ \rangle,\\ \left\langle\begin{array}{ll}\texttt{index}-1 & \texttt{value}-\texttt{Dist}_{i_1},\\ \texttt{index}-2 & \texttt{value}-\texttt{Dist}_{i_2},\\ & \vdots\\ \texttt{index}-m & \texttt{value}-\texttt{Dist}_{i_m}\end{array}\right\rangle\end{array}\right)$$

where:

- $\texttt{Next}_i$ is a domain variable that gives the index of the location the vehicle will visit just after location i,

- $\mathtt{distance}_i$ is a domain variable that corresponds to the distance between location i and the location the vehicle will visit just after,
- $\mathtt{Dist}_{i_1}, \mathtt{Dist}_{i_2}, \ldots, \mathtt{Dist}_{i_m}$ are the respective distances between location i and locations $1, 2, \ldots, m$.

An other example where the table argument corresponds to domain variables is described in the keyword entry assignment to the same set of values.

Remark

Originally, the parameters of the elem constraint had the form element(INDEX, TABLE, VALUE), where INDEX and VALUE were two domain variables and TABLE was a list of non-negative integers.

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 elem constraint. We use the reformulation $I = J - p + 1 \wedge$ elem($\langle$index $- I$ value $- V\rangle$, TABLE), where I and J are domain variables respectively ranging from 1 to |TABLE| and from p to $p +$ |TABLE| $- 1$.

Systems

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

See also

common keyword: elem_from_to, element_matrix, element_product, element_sparse *(array constraint)*, elements_sparse, stage_element *(data constraint)*.

implied by: element.

implies: element *(single item replaced by two variables)*, element_greatereq, element_lesseq, elements.

system of constraints: elements.

uses in its reformulation: elements_alldifferent.

Keywords

characteristic of a constraint: automaton, automaton without counters, reified automaton constraint.

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.

Cond. implications

elem(ITEM, TABLE)
with TABLE.value ≥ 0
implies bin_packing_capa(TABLE, ITEM).

Arc input(s)	ITEM TABLE
Arc generator	*PRODUCT* $\mapsto$ collection(item, table)
Arc arity	2
Arc constraint(s)	• item.index = table.index • item.value = table.value
Graph property(ies)	**NARC** = 1

Graph model We regroup the INDEX and VALUE parameters of the original element constraint element(INDEX, TABLE, VALUE) into the parameter ITEM. We also make explicit the different indices of the table TABLE.

Parts (A) and (B) of Figure 5.296 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.

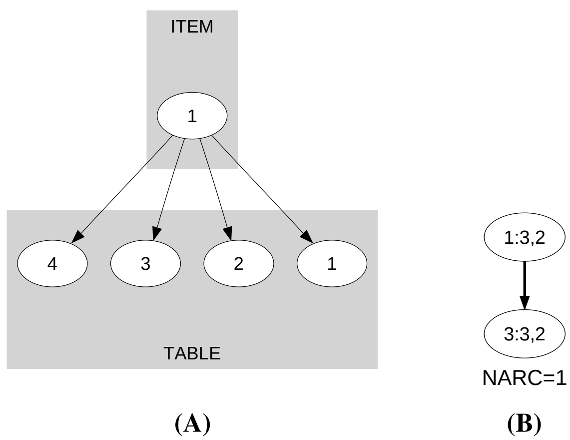


Figure 5.296: Initial and final graph of the elem constraint

Signature Since all the index attributes of TABLE are distinct and because of the first condition of the arc constraint the final graph cannot have more than one arc. Therefore we can rewrite $\mathbf{NARC} = 1$ to $\mathbf{NARC} \geq 1$ and simplify $\underline{\overline{\mathbf{NARC}}}$ to $\overline{\mathbf{NARC}}$.

Automaton Figure 5.297 depicts the automaton associated with the `elem` constraint. Let `INDEX` and `VALUE` respectively be the `index` and the `value` attributes of the unique item of the `ITEM` collection. Let $\texttt{INDEX}_i$ and $\texttt{VALUE}_i$ respectively be the `index` and the `value` attributes of item i of the `TABLE` collection. To each quadruple $(\texttt{INDEX}, \texttt{VALUE}, \texttt{INDEX}_i, \texttt{VALUE}_i)$ corresponds a 0-1 signature variable S_i as well as the following signature constraint: $((\texttt{INDEX} = \texttt{INDEX}_i) \wedge (\texttt{VALUE} = \texttt{VALUE}_i)) \Leftrightarrow S_i$.

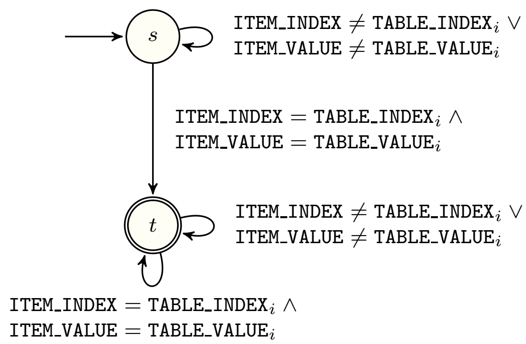


Figure 5.297: Automaton of the $\texttt{elem}(\texttt{ITEM}, \texttt{TABLE})$ constraint (once one finds the right item – index and value – in the table, one switches from the initial state s to the accepting state t)

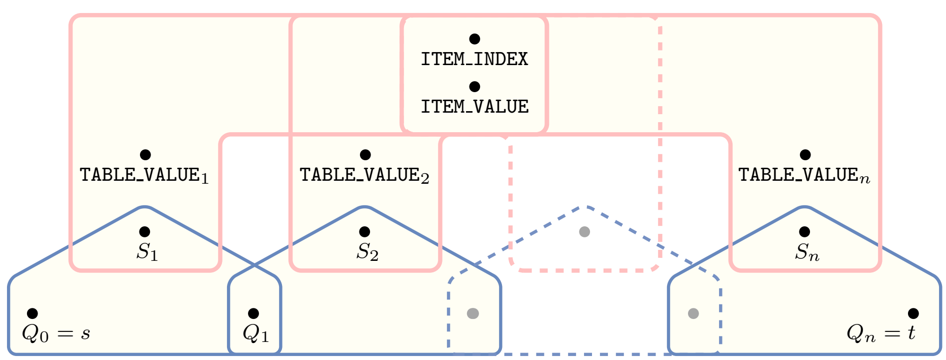


Figure 5.298: Hypergraph of the reformulation corresponding to the automaton of the `elem` constraint