5.196  int_value_precede_chain

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>LINKS</th>
<th>AUTOMATON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>[258]</td>
<td></td>
</tr>
<tr>
<td>Constraint</td>
<td>int_value_precede_chain(VALUES, VARIABLES)</td>
<td></td>
</tr>
<tr>
<td>Synonyms</td>
<td>precede, precedence, value_precede_chain.</td>
<td></td>
</tr>
<tr>
<td>Arguments</td>
<td>VALUES : collection(var−int) VARIABLES : collection(var−dvar)</td>
<td></td>
</tr>
<tr>
<td>Restrictions</td>
<td>required(VALUES, var) distinct(VALUES, var) required(VARIABLES, var)</td>
<td></td>
</tr>
</tbody>
</table>

**Purpose**

Assuming $n$ denotes the number of items of the VALUES collection, the following condition holds for every $i \in [1, n - 1]$: When it is defined, the first occurrence of the $(i + 1)^{th}$ value of the VALUES collection should be preceded by the first occurrence of the $i^{th}$ value of the VALUES collection.

**Example**

\[ ((4, 0, 1), (4, 0, 6, 1, 0)) \]

The int_value_precede_chain constraint holds since within the sequence 4, 0, 6, 1, 0:

- The first occurrence of value 4 occurs before the first occurrence of value 0.
- The first occurrence of value 0 occurs before the first occurrence of value 1.

**Typical**

\[ |VALUES| > 1 \]  
strictly_increasing(VALUES)  
|VARIABLES| > |VALUES|  
range(VARIABLES.var) > 1  
used_by(VARIABLES, VALUES)

**Symmetry**

An occurrence of a value of VARIABLES.var that does not occur in VALUES.var can be replaced by any other value that also does not occur in VALUES.var.

**Arg. properties**

- Contractible wrt. VALUES.
- Suffix-contractible wrt. VARIABLES.
- Aggregate: VALUES(id), VARIABLES(union).  


Usage

The `int_value_precede_chain` constraint is useful for breaking symmetries in graph colouring problems. We set a `int_value_precede_chain` constraint on all variables $V_1, V_2, \ldots, V_n$ associated with the vertices of the graph to colour, where we state that the first occurrence of colour $i$ should be located before the first occurrence of colour $i+1$ within the sequence $V_1, V_2, \ldots, V_n$.

Figure 5.437 illustrates the problem of colouring earth and mars from Thom Sulanke. Part (A) of Figure 5.437 provides a solution where the first occurrence of each value of $i, (i \in \{1, 2, \ldots, 8\})$ is located before the first occurrence of value $i+1$. This is obtained by using the following constraints:

$$\begin{align*}
A &\not= B, A &\not= E, A &\not= F, A &\not= C, A &\not= H, A &\not= I, A &\not= J, A &\not= K, \\
B &\not= A, B &\not= C, B &\not= F, B &\not= G, B &\not= H, B &\not= I, B &\not= J, B &\not= K, \\
C &\not= B, C &\not= D, C &\not= F, C &\not= G, C &\not= H, C &\not= I, C &\not= J, C &\not= K, \\
D &\not= C, D &\not= E, D &\not= F, D &\not= G, D &\not= H, D &\not= I, D &\not= J, D &\not= K, \\
E &\not= A, E &\not= D, E &\not= F, E &\not= G, E &\not= H, E &\not= I, E &\not= J, E &\not= K, \\
F &\not= A, F &\not= B, F &\not= C, F &\not= D, F &\not= E, F &\not= G, F &\not= H, F &\not= I, F &\not= J, F &\not= K, \\
G &\not= A, G &\not= B, G &\not= C, G &\not= D, G &\not= E, G &\not= F, G &\not= H, G &\not= I, G &\not= J, G &\not= K, \\
K &\not= A, K &\not= B, K &\not= C, K &\not= D, K &\not= E, K &\not= F, K &\not= G, K &\not= H, K &\not= I, K &\not= J, K.
\end{align*}$$

Part (B) provides a symmetric solution where the value precedence constraints between the pair of values $(1, 2), (2, 3), (4, 5), (7, 8)$ and $(8, 9)$ are all violated (each violation is depicted by a dashed arc).

Remark

When we have more than one class of interchangeable values (i.e., a partition of interchangeable values) we can use one `int_value_precede_chain` constraint for breaking value symmetry in each class of interchangeable values. However it was shown in [439] that enforcing arc-consistency for such a conjunction of `int_value_precede_chain` constraints is NP-hard.

Algorithm

The 2004 reformulation [28] associated with the automaton of the Automaton slot achieves arc-consistency since the corresponding constraint network is a Berge-acyclic constraint network. Later on, another formulation into a sequence of ternary sliding constraints was proposed by [438]. It also achieves arc-consistency for the same reason.

Systems

`precede` in Gecode, `value_precede_chain` in MiniZinc.

See also

specialisation: `int_value_precede` (sequence of at least 2 values replaced by sequence of 2 values).

Keywords

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

constraint network structure: Berge-acyclic constraint network.

constraint type: order constraint.
Figure 5.437: Using the int_value_precede_chain constraint for breaking symmetries in graph colouring problems; there is an arc between the first occurrence of value $v \ (1 \leq v \leq 8)$ in the sequence of variables A, B, C, D, E, F, G, H, I, J, K, and the first occurrence of value $v + 1$ (a plain arc if the corresponding value precedence constraint holds, a dashed arc otherwise)

filtering: arc-consistency.
problems: graph colouring.
symmetry: symmetry, indistinguishable values, value precedence.
Automaton

Figure 5.438 depicts the automaton associated with the `int_value_precede_chain` constraint. Let $n$ and $m$ respectively denote the number of variables of the `VARIABLES` collection and the number of values of the `VALUES` collection. Let $\text{VAR}_i$ be the $i^{th}$ variable of the `VARIABLES` collection. Let $\text{val}_v$ ($1 \leq v \leq m$) denote the $v^{th}$ value of the `VALUES` collection.

$\text{VAR}_i = \text{val}_1$

$\text{VAR}_i = \text{val}_2$

$\text{VAR}_i = \text{val}_1 \lor \text{VAR}_i = \text{val}_2$

$\text{VAR}_i = \text{val}_m$

$\text{VAR}_i = \text{val}_1 \lor \cdots \lor \text{VAR}_i = \text{val}_m$

Figure 5.439: Hypergraph of the reformulation corresponding to the automaton of the `int_value_precede_chain` constraint (state $s_i$ means that (1) each value $\text{val}_1, \text{val}_2, \ldots, \text{val}_i$ was already encountered at least once, and that (2) value $\text{val}_{i+1}$ was not yet found)
We now show how to construct such an automaton systematically. For this purpose let us first introduce some notations:

- Without loss of generality we assume that we have at least two values (i.e., \( m \geq 2 \)).
- Let \( C \) be the set of values that can be potentially assigned to a variable of the \texttt{VARIABLES} collection, but which do not belong to the values of the \texttt{VALUES} collection (i.e., \( C = (\text{dom} (\text{VAR}_1) \cup \text{dom} (\text{VAR}_2) \cup \cdots \cup \text{dom} (\text{VAR}_n) - \{\text{val}_1, \text{val}_2, \ldots, \text{val}_m\} = \{w_1, w_2, \ldots, w_{|C|}\} \)).

The states and transitions of the automaton are respectively defined in the following way:

- We have \( m + 1 \) states labelled \( s_0, s_1, \ldots, s_m \) from which \( s_0 \) is the initial state. All states are accepting states.
- We have the following three sets of transitions:
  1. For all \( v \in [0, m - 1] \), a transition from \( s_v \) to \( s_{v+1} \) labelled by value \( \text{val}_{v+1} \). Each transition of this type will be triggered on the first occurrence of value \( \text{val}_{v+1} \) within the variables of the \texttt{VARIABLES} collection.
  2. For all \( v \in [1, m] \) and for all \( w \in [1, v] \), a self loop on \( s_v \) labelled by value \( \text{val}_w \). Such transitions encode the fact that we stay in the same state as long as we have a value that was already encountered.
  3. If the set \( C \) is not empty, then for all \( v \in [0, m] \) a self loop on \( s_v \) labelled by the fact that we take a value not in \texttt{VALUES} (i.e., a value in \( C \)). This models the fact that, encountering a value that does not belong to the set of values of the \texttt{VALUES} collection, leaves us in the same state.