5.196 int_value_precede_chain

DESCRIPTION LINKS AUTOMATON

Origin [258]

Constraint `int_value_precede_chain(VALUES, VARIABLES)`

Synonyms `precede`, `precedence`, `value_precede_chain`.

Arguments

```
VALUES    : collection(var-int)
VARIABLES : collection(var-dvar)
```

Restrictions

```
required(VALUES, var)
distinct(VALUES, var)
required(VARIABLES, var)
```

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

$(\langle 4, 0, 1\rangle, \langle 4, 0, 6, 1, 0\rangle)$

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

$|\texttt{VALUES}| > 1$
`strictly_increasing(VALUES)`
$|\texttt{VARIABLES}| > |\texttt{VALUES}|$
$\texttt{range(VARIABLES.var)} > 1$
`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 5.454 illustrates the problem of *colouring earth and mars* from Thom Sulanke. Part (A) of Figure 5.454 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:

$$\left\{\begin{array}{l}
\mathtt{A} \neq \mathtt{B}, \mathtt{A} \neq \mathtt{E}, \mathtt{A} \neq \mathtt{F}, \mathtt{A} \neq \mathtt{G}, \mathtt{A} \neq \mathtt{H}, \mathtt{A} \neq \mathtt{I}, \mathtt{A} \neq \mathtt{J}, \mathtt{A} \neq \mathtt{K}, \\
\mathtt{B} \neq \mathtt{A}, \mathtt{B} \neq \mathtt{C}, \mathtt{B} \neq \mathtt{F}, \mathtt{B} \neq \mathtt{G}, \mathtt{B} \neq \mathtt{H}, \mathtt{B} \neq \mathtt{I}, \mathtt{B} \neq \mathtt{J}, \mathtt{B} \neq \mathtt{K}, \\
\mathtt{C} \neq \mathtt{B}, \mathtt{C} \neq \mathtt{D}, \mathtt{C} \neq \mathtt{F}, \mathtt{C} \neq \mathtt{G}, \mathtt{C} \neq \mathtt{H}, \mathtt{C} \neq \mathtt{I}, \mathtt{C} \neq \mathtt{J}, \mathtt{C} \neq \mathtt{K}, \\
\mathtt{D} \neq \mathtt{C}, \mathtt{D} \neq \mathtt{E}, \mathtt{D} \neq \mathtt{F}, \mathtt{D} \neq \mathtt{G}, \mathtt{D} \neq \mathtt{H}, \mathtt{D} \neq \mathtt{I}, \mathtt{D} \neq \mathtt{J}, \mathtt{D} \neq \mathtt{K}, \\
\mathtt{E} \neq \mathtt{A}, \mathtt{E} \neq \mathtt{D}, \mathtt{E} \neq \mathtt{F}, \mathtt{E} \neq \mathtt{G}, \mathtt{E} \neq \mathtt{H}, \mathtt{E} \neq \mathtt{I}, \mathtt{E} \neq \mathtt{J}, \mathtt{E} \neq \mathtt{K}, \\
\mathtt{F} \neq \mathtt{A}, \mathtt{F} \neq \mathtt{B}, \mathtt{F} \neq \mathtt{C}, \mathtt{F} \neq \mathtt{D}, \mathtt{F} \neq \mathtt{E}, \mathtt{F} \neq \mathtt{G}, \mathtt{F} \neq \mathtt{H}, \mathtt{F} \neq \mathtt{I}, \mathtt{F} \neq \mathtt{J}, \mathtt{F} \neq \mathtt{K}, \\
\mathtt{G} \neq \mathtt{A}, \mathtt{G} \neq \mathtt{B}, \mathtt{G} \neq \mathtt{C}, \mathtt{G} \neq \mathtt{D}, \mathtt{G} \neq \mathtt{E}, \mathtt{G} \neq \mathtt{F}, \mathtt{G} \neq \mathtt{H}, \mathtt{G} \neq \mathtt{I}, \mathtt{G} \neq \mathtt{J}, \mathtt{G} \neq \mathtt{K}, \\
\mathtt{H} \neq \mathtt{A}, \mathtt{H} \neq \mathtt{B}, \mathtt{H} \neq \mathtt{C}, \mathtt{H} \neq \mathtt{D}, \mathtt{H} \neq \mathtt{E}, \mathtt{H} \neq \mathtt{F}, \mathtt{H} \neq \mathtt{G}, \mathtt{H} \neq \mathtt{I}, \mathtt{H} \neq \mathtt{J}, \mathtt{H} \neq \mathtt{K}, \\
\mathtt{I} \neq \mathtt{A}, \mathtt{I} \neq \mathtt{B}, \mathtt{I} \neq \mathtt{C}, \mathtt{I} \neq \mathtt{D}, \mathtt{I} \neq \mathtt{E}, \mathtt{I} \neq \mathtt{F}, \mathtt{I} \neq \mathtt{G}, \mathtt{I} \neq \mathtt{H}, \mathtt{I} \neq \mathtt{J}, \mathtt{I} \neq \mathtt{K}, \\
\mathtt{J} \neq \mathtt{A}, \mathtt{J} \neq \mathtt{B}, \mathtt{J} \neq \mathtt{C}, \mathtt{J} \neq \mathtt{D}, \mathtt{J} \neq \mathtt{E}, \mathtt{J} \neq \mathtt{F}, \mathtt{J} \neq \mathtt{G}, \mathtt{J} \neq \mathtt{H}, \mathtt{J} \neq \mathtt{I}, \mathtt{J} \neq \mathtt{K}, \\
\mathtt{K} \neq \mathtt{A}, \mathtt{K} \neq \mathtt{B}, \mathtt{K} \neq \mathtt{C}, \mathtt{K} \neq \mathtt{D}, \mathtt{K} \neq \mathtt{E}, \mathtt{K} \neq \mathtt{F}, \mathtt{K} \neq \mathtt{G}, \mathtt{K} \neq \mathtt{H}, \mathtt{K} \neq \mathtt{I}, \mathtt{K} \neq \mathtt{J}, \\
\mathtt{int_value_precede_chain}(\langle 1,2,3,4,5,6,7,8,9\rangle, \langle \mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D},\mathtt{E},\mathtt{F},\mathtt{G},\mathtt{H},\mathtt{I},\mathtt{J},\mathtt{K}\rangle).
\end{array}\right.$$

Part (B) provides a symmetric solution where the value precedence constraints between the pairs 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

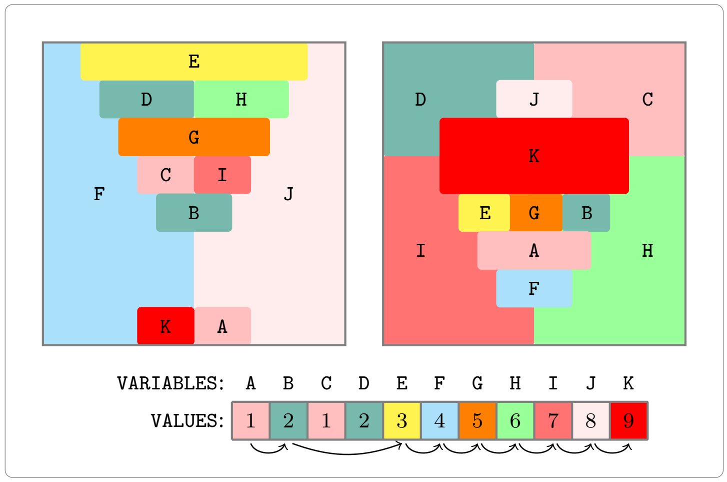
specialisation: int_value_precede *(*sequence *of at least* 2 values *replaced by* sequence *of* 2 values*)*.

Keywords

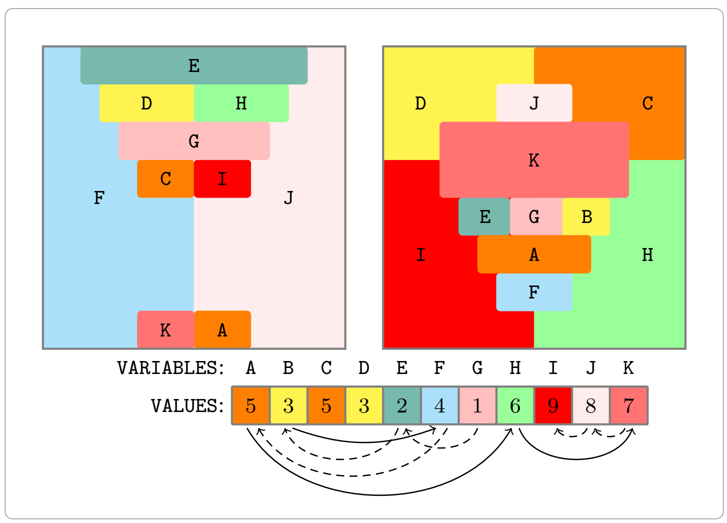
characteristic of a constraint: automaton, automaton without counters, reified automaton constraint.

constraint network structure: Berge-acyclic constraint network.

constraint type: order constraint.



(A) Chain of satisfied value precedences between first occurrence of consecutive values



(B) Broken value precedences between first occurrence of consecutive values (in dashed)

Figure 5.454: 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 5.455 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 $\mathtt{VAR}_i$ be the i^{th} variable of the `VARIABLES` collection. Let $\mathtt{val}_v$ $(1 \leq v \leq m)$ denote the v^{th} value of the `VALUES` collection.

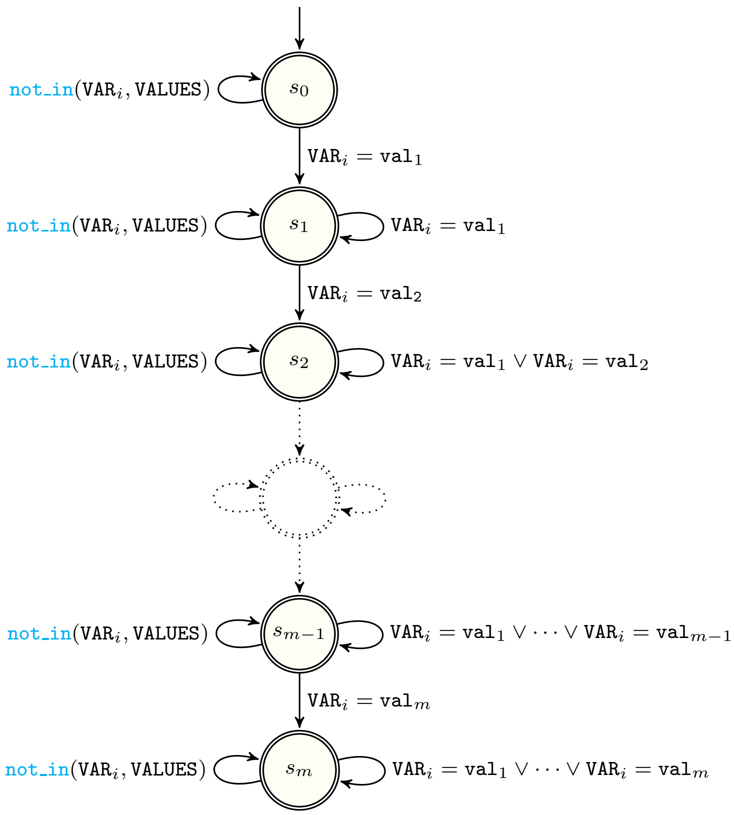


Figure 5.455: Automaton of the `int_value_precede_chain` constraint (state s_i means that (1) each value $\mathtt{val}_1, \mathtt{val}_2, \ldots, \mathtt{val}_i$ was already encountered at least once, and that (2) value $\mathtt{val}_{i+1}$ was not yet found)

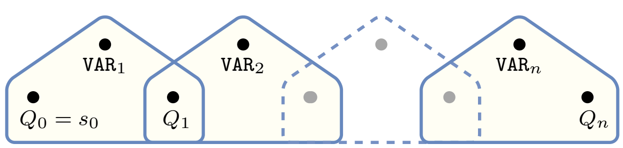


Figure 5.456: Hypergraph of the reformulation corresponding to the automaton of the `int_value_precede_chain` constraint (since all states of the automaton are accepting there is no restriction on the last variable Q_n)

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 $\mathcal{C}$ be the set of values that can be potentially assigned to a variable of the VARIABLES collection, but which do not belong to the values of the VALUES collection (i.e., $\mathcal{C} = (dom(\mathtt{VAR}_1) \cup dom(\mathtt{VAR}_2) \cup \cdots \cup dom(\mathtt{VAR}_n) - \{\mathtt{val}_1, \mathtt{val}_2, \ldots, \mathtt{val}_m\} = \{w_1, w_2, \ldots, w_{|\mathcal{C}|}\}$.

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 $\mathtt{val}_{v+1}$. Each transition of this type will be triggered on the first occurrence of value $\mathtt{val}_{v+1}$ within the variables of the 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 $\mathtt{val}_w$. 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 $\mathcal{C}$ 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 VALUES (i.e., a value in $\mathcal{C}$). This models the fact that, encountering a value that does not belong to the set of values of the VALUES collection, leaves us in the same state.