5.317 pattern

DESCRIPTION LINKS AUTOMATON

Origin [83]

Constraint pattern(VARIABLES, PATTERNS)

Type PATTERN : collection(var−int)

Arguments
VARIABLES : collection(var−dvar)
PATTERNS : collection(pat − PATTERN)

Restrictions
required(PATTERN, var)
PATTERN.var ≥ 0
change(0, PATTERN, =)
$|$PATTERN$| > 1$
required(VARIABLES, var)
required(PATTERNS, pat)
$|$PATTERNS$| > 0$
same_size(PATTERNS, pat)

Purpose

We quote the definition from the original article [83, page 157] introducing the pattern constraint:

> "*We call a k-pattern ($k > 1$) any sequence of k elements such that no two successive elements have the same value. Consider a set $V = \{v_1, v_2, \ldots, v_m\}$ and a sequence* s $= s_1\ s_2\ \ldots\ s_n$ *of elements of V. In this context, a* stretch *is a maximum subsequence of variables of* s *which all have the same value. Consider now the sequence $v_{i1}\ v_{i2}\ \ldots\ v_{il}$ of the types of the successive stretches that appear in* s*. Let $\mathcal{P}$ be a set of k-patterns.* s *satisfies $\mathcal{P}$ if and only if every subsequence of k elements in $v_{i1}\ v_{i2}\ \ldots, v_{il}$ belongs to $\mathcal{P}$.*"

Example

$$\left(\begin{array}{l}\langle 1,1,2,2,2,1,3,3\rangle,\\ \langle \texttt{pat}-\langle 1,2,1\rangle, \texttt{pat}-\langle 1,2,3\rangle, \texttt{pat}-\langle 2,1,3\rangle\rangle\end{array}\right)$$

The pattern constraint holds since, as depicted by Figure 5.641, all its sequences of three consecutive stretches correspond to one of the 3-patterns given in the PATTERNS collection.

Typical
$|$VARIABLES$| > 2$
range(VARIABLES.var) > 1

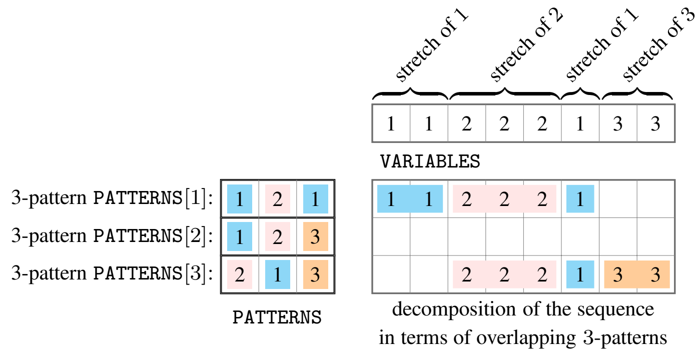

Figure 5.641: The sequence of the **Example** slot, its four stretches and the corresponding two 3-patterns 1 2 1 and 2 1 3

Symmetries

- Items of `PATTERNS` are permutable.
- Items of `VARIABLES` and `PATTERNS.pat` are simultaneously reversable.
- All occurrences of two distinct tuples of values in `VARIABLES.var` or `PATTERNS.pat.var` can be swapped; all occurrences of a tuple of values in `VARIABLES.var` or `PATTERNS.pat.var` can be renamed to any unused tuple of values.

Arg. properties

- Prefix-contractible wrt. `VARIABLES`.
- Suffix-contractible wrt. `VARIABLES`.

Usage

The `pattern` constraint was originally introduced within the context of staff scheduling. In this context, the value of the i^{th} variable of the `VARIABLES` collection corresponds to the type of shift performed by a person on the i^{th} day. A *stretch* is a maximum sequence of consecutive variables that are all assigned to the same value. The `pattern` constraint imposes that each sequence of k consecutive stretches belongs to a given list of patterns.

Remark

A generalisation of the `pattern` constraint to the `regular` constraint enforcing the fact that a sequence of variables corresponds to a regular expression is presented in [306].

See also

common keyword: `group` *(timetabling constraint)*, `sliding_distribution` *(sliding sequence constraint)*, `stretch_circuit`, `stretch_path` *(sliding sequence constraint,timetabling constraint)*, `stretch_path_partition` *(sliding sequence constraint)*.

Keywords

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

constraint network structure: Berge-acyclic constraint network.

constraint type: timetabling constraint, sliding sequence constraint.

filtering: arc-consistency.

Automaton Taking advantage that all k-patterns have the same length k, it is straightforward to construct an automaton that only accepts solutions of the `pattern` constraint. Figure 5.642 depicts the automaton associated with the `pattern` constraint of the **Example** slot. The construction can be done in three steps:

- First, build a prefix tree of all the k-patterns. In the context of our example, this gives all arcs of Figure 5.642, except self loops and the arc from s_3 to s_7.
- Second, find out the transitions that exit a leave of the tree. For this purpose we remove the first symbol of the corresponding k-pattern, add at the end of the remaining k-pattern a symbol corresponding to a stretch value, and check whether the new pattern belongs or not to the set of k-patterns of the `pattern` constraint. When the new pattern belongs to the set of k-patterns we add a corresponding transition. For instance, in the context of our example, consider the leave s_3 that is associated with the 3-pattern $1, 2, 1$. We remove the first symbol 1 and get $2, 1$. We then try to successively add the stretch values 1, 2 and 3 to the end of $2, 1$ and check if the corresponding patterns $2, 1, 1$, $2, 1, 2$ and $2, 1, 3$ belong or not to our set of 3-patterns. Since only $2, 1, 3$ is a 3-pattern we add a new transition between the corresponding leaves of the prefix tree (i.e., a transition from s_3 to s_7).
- Third, in order to take into account that each value of a k-pattern corresponds in fact to a given stretch value (i.e., several consecutive values that are assigned the same value), we add a self loop to all non-source states with a transition label that corresponds to the transition label of their entering arc.

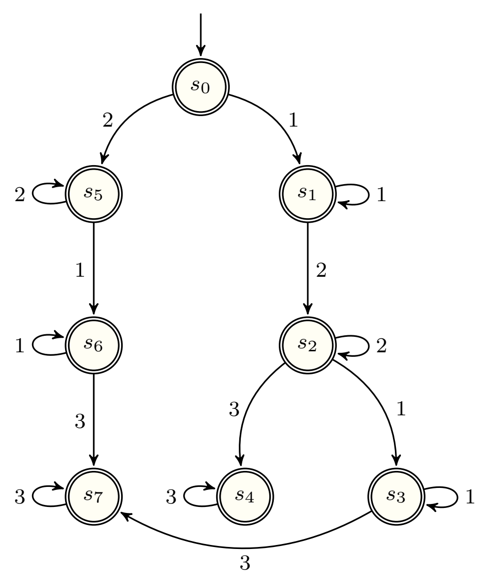

Figure 5.642: Automaton of the `pattern` constraint of the **Example** slot