5.386 sum_free

DESCRIPTION LINKS

Origin [428]

Constraint `sum_free(S)`

Argument `S : svar`

Purpose Impose for all pairs of values (not necessarily distinct) i, j of the set `S` the fact that the sum $i + j$ is not an element of `S`.

Example $(\{1, 3, 5, 9\})$

The `sum_free`$(\{1, 3, 5, 9\})$ constraint holds since:

- $1 + 1 = 2 \notin$ S, $1 + 3 = 4 \notin$ S, $1 + 5 = 6 \notin$ S, $1 + 9 = 10 \notin$ S.
- $3 + 3 = 6 \notin$ S, $3 + 5 = 8 \notin$ S, $3 + 9 = 12 \notin$ S.
- $5 + 5 = 10 \notin$ S, $5 + 9 = 14 \notin$ S.

Usage The `sum_free` constraint was introduced by W.-J. van Hoeve and A. Sabharwal in order to model in a concise way Schur problems.

- On one hand, the first model has n domain variables x_i $(1 \leq i \leq n)$, where x_i corresponds to the subset in which element i occurs. The constraints $x_i = s \wedge x_j = s \Rightarrow x_{i+j} \neq s$ $(s \in [1, k], i, j \in [1, n], i \leq j, i + j \leq n)$ enforce that the k subsets are sum-free. We have $O(k \cdot n^2)$ such constraints.
- On the other hand, the model proposed by W.-J. van Hoeve and A. Sabharwal represents in an explicit way with a set variable S_i $(1 \leq i \leq n)$ each subset of the partition we are looking for. Now, to express the fact that these k subsets are sum-free they simply use k `sum_free` constraints of the form `sum_free`(S_i).

While the two models have the same behaviour when we focus on the number of backtracks the second model is much more efficient from a memory point of view.

Algorithm W.-J. van Hoeve and A. Sabharwal have proposed an algorithm that enforces bound-consistency for the `sum_free` constraint in [428].

Keywords **constraint arguments:** unary constraint, constraint involving set variables.

constraint type: predefined constraint.

filtering: bound-consistency.

problems: Schur number.