PVS-Type Definitions

Institute for Software & Systems Engineering

```
type ConstraintRelationships = PVSType<bool, set of 1..nScs> =
  params {
    array[int, 1..2] of 1..nScs: crEdges; % adjacency matrix
    bool: useSPD;
  } in
  instantiates with "../mbr_types/cr_type.mzn" {
    times -> link_invert_booleans;
    is_worse -> is_worse_cr;
    top -> {};
  };
```

- `PVSType<S,E>` distinguishes
  - Specification type S
  - Element type E
- Combination operation: `times` : $S^n \rightarrow E$
- Ordering relation: `isWorse` $\subseteq E \times E$

PVS-Types

```
type WeightedCsp = PVSType<bool, int> =
  params {
    int: k;
    array[1..nScs] of 1..k: weights :: default('1');
  } in
  instantiates with "../mbr_types/weighted_type.mzn" {
    times -> weighted_sum;
    is_worse -> is_worse_weighted;
    top -> 0;
  };

type CostFunctionNetwork = PVSType<0..k> =
  params {
    int: k :: default('1000');
  } in instantiates with "../mbr_types/cfn_type.mzn" {
    times -> sum;
    is_worse -> is_worse_weighted;
    top -> 0;
 };
```

PVS-Instantiation for Weighted-CSp

Institute for
Software & Systems
Engineering

```
PVS: cr1 = new WeightedCsp("cr1") {
   soft-constraint c1: 'x + 1 = y' :: weights('2');
   soft-constraint c2: 'z = y + 2' :: weights('1');
   soft-constraint c3: 'x + y <= 3' :: weights('1');

   k : '20';
};
```

- Weights can be annotated
- Or passed as array (`[2,1,1]`)