Constraint Programming
Prof. Dr. Thom Frühwirth

assignment #5

---

## Warmup

**Exercise 1** (Cardinality Constraints)**.**
Extend `boole.pl` (from assignment #4) to handle cardinality constraints `card/4` with semantics given in the lecture.

a) Implement the rules together with the required auxiliary predicates.
b) Introduce a constraint `labeling` together with appropriate rule(s) to label variables.
c) Cardinality constraints can be combined with the existing Boolean constraints, e.g.

```
card2and  @ card(0,1,[X,Y],2) <=> and(X,Y,0).
card2neg  @ card(1,1,[X,Y],2) <=> neg(X,Y).
```

Find similar rules for (at least) xor and nand.

## Constraint-system Rational Tree

**Exercise 2.** Implement the CHR-constraint `X eq Y` that succeeds iff $CET \models$ `X`$\doteq$`Y`.
Clark's equality theory $CET$ should be coded "naturally", i.e., implement the axioms as propagation rules (whenever possible).
**Hints:**

- `f(X1,...,XN)=..[f|X1,...,XN]`
- Rules leading to immediate contradiction should go first in the program text.
- For termination reasons pay attention not to have multiple copies of a constraint in the store.

**Queries:** Unification examples from assignment #1.

Extend your implementation, s.t. queries like `X eq f(Y), Y eq f(X)` can be treated (occur-check). A simple solution introduces one (or several) rule(s) for variable-substitution.

**Exercise 3.** The constraint theory $CT$ should define the (purely) syntactic inequality $\dot{\neq}$ between two terms along the lines of $CET$:

| | |
|---|---|
| *irreflexivity* | $\forall(x \dot{\neq} x \rightarrow \bot)$ |
| *symmetry* | $\forall(x \dot{\neq} y \rightarrow y \dot{\neq} x)$ |
| *compatibility* | $\forall(x_1 \dot{\neq} y_1 \vee \ldots \vee x_n \dot{\neq} y_n \rightarrow f(x_1,\ldots,x_n) \dot{\neq} f(y_1,\ldots,y_n))$ |
| *decomposition* | $\forall(f(x_1,\ldots,x_n) \dot{\neq} f(y_1,\ldots,y_n) \rightarrow x_1 \dot{\neq} y_1 \vee \ldots \vee x_n \dot{\neq} y_n)$ |
| *distinctness* | $\forall(\top \rightarrow f(x_1,\ldots,x_n) \dot{\neq} g(y_1,\ldots,y_m))$ iff $f \neq g$ or $n \neq m$ |
| *cylicity* | $\forall(\top \rightarrow x \dot{\neq} t)$ if $t$ is not a variable and $x$ appears in $t$ |

The to be implemented CHR-constraint `X neq Y` should succeed iff $CT \models$ `X`$\dot{\neq}$`Y`.
Use the RT-solver implementation from the lecture as blueprint for your implementation. Disjunction, needed for *compatibility* and *decomposition*, should be implemented by a CHR$^\vee$ constraint `one_neq/2` as negated `same_args/` constraint. The two arguments of `one_neq/2` are lists of same length and the CHR$^\vee$ should succeed iff at least on pair of list-elements is unequal.

**Note:** Using disjunction in CHR$^\vee$-bodies requires a (mandatory) guard in SICStus Prolog:
`rule @ Head <=> true | (Goal1 ; Goal2).`

**Queries:**

(1) `?- X neq f(X)`
(2) `?- f(a,X) neq f(X,Y)`
(3) `?- f(g(X),a) neq f(Y,X)`