Warmup

Exercise 1 (Cardinality Constraints).
Extend boole.pl (from assignment #8) 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 \( CET \models x \neq y \).
Clark’s equality theory \( CET \) should be coded “naturally”, i.e., implement the axioms as propagation rules (whenever possible).

Hints:
- \( f(X_1, \ldots, X_N) = f[Y_1, \ldots, Y_N] \)
- 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.

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

irreflexivity \( \forall (x \neq x \rightarrow \bot) \)
symmetry \( \forall (x \neq y \rightarrow y \neq x) \)
compatibility \( \forall (x_1 \neq y_1 \lor \ldots \lor x_n \neq y_n \rightarrow f(x_1, \ldots, x_n) \neq f(y_1, \ldots, y_n)) \)
decomposition \( \forall (f(x_1, \ldots, x_n) \neq f(y_1, \ldots, y_n) \rightarrow x_1 \neq y_1 \lor \ldots \lor x_n \neq y_n) \)
distinctness \( \forall (\top \rightarrow f(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_m)) \) if \( f \neq g \) or \( n \neq m \)
cyclic \( \forall (\top \rightarrow x \neq t) \) if \( t \) is not a variable and \( x \) appears in \( t \)

The to be implemented CHR-constraint \( X \neq Y \) should succeed iff \( CT \models x \neq y \).
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\(^V\) constraint one_neq/2 as negated same_args/ constraint. The two arguments of one_neq/2 are lists of same length and the CHR\(^V\) should succeed iff at least on pair of list-elements is unequal.

Note: Using disjunction in CHR\(^V\)-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)