Constraint Programmierung
Prof. Dr. Thom Frühwirth, Marc Meister

WS4 – Blatt 10
Besprechung am 25. Jan., 14 Uhr, O27/2201
Veranstaltungsseite `http://www.informatik.uni-ulm.de/pm/index.php?id=82`

Constraintsystem B
Verwenden Sie den Constraintlöser `boole.pl` von Übungsblatt 9.

Aufgabe 29 (Fehleranalyse in digitalen Schaltungen)**.**
Schreiben Sie ein CHR-Programm `faultanalysis(I1,I2,I3,O1,O2,F1,F2,F3,F4,F5)`, das den (maximal einen) defekten Gatter eines Volladdierers feststellt. Verwenden Sie die in der Vorlesung vorgestellte Modellierung. Testen Sie mit folgenden Anfragen:

```
faultanalysis(0,0,1,0,1,F1,F2,F3,F4,F5).
faultanalysis(1,1,0,0,0,F1,F2,F3,F4,F5).
```

Aufgabe 30 (Cardinality Constraints)**.**
Extend `boole.pl` 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 similiar rules for (at least) xor and nand.

Constraintsystem Rational Tree

Aufgabe 31. Schreiben Sie ein CHR-Constraint `X eq Y`, das zutrifft, wenn $CET \models$ `X`$\doteq$`Y`.
Implementieren Sie die Gleichheitstheorie CET nach Clark möglichst „naturgetreu", d.h. mittels ihrer Axiome als Propagierungsregeln, soweit dies möglich ist.
Hinweise:
Den Funktionausdruck `f(X)` zerlegt `f(X)=..[F|A]` in `F = f` und `A = [X]`.
Regeln, die unmittelbar zum Widerspruch führen, sollten als erstes im Programmtext stehen.
Achten Sie zwecks Terminierung darauf, Constraints nicht mehrfach im Speicher zu haben.
Testziele: Unifikationsbeispiele von Blatt 1.

Erweitern Sie, so daß Anfragen wie `X eq f(Y), Y eq f(X)` bearbeitet werden können, indem Sie eine oder mehrere Regeln für Variablen-Substitution einführen.

Aufgabe 32. Gegeben sei eine Constrainttheorie CT, die die rein syntaktische Ungleichheit $\dot{\neq}$ zwischen zwei Termen analog zu CET definiert:

Irreflexivität	$\forall(x \dot{\neq} x \rightarrow \bot)$
Symmetrie	$\forall(x \dot{\neq} y \rightarrow y \dot{\neq} x)$
Verträglichkeit	$\forall(x_1 \dot{\neq} y_1 \vee \ldots \vee x_n \dot{\neq} y_n \rightarrow f(x_1,\ldots,x_n) \dot{\neq} f(y_1,\ldots,y_n))$
Zerlegung	$\forall(f(x_1,\ldots,x_n) \dot{\neq} f(y_1,\ldots,y_n) \rightarrow x_1 \dot{\neq} y_1 \vee \ldots \vee x_n \dot{\neq} y_n)$
Verschiedenheit	$\forall(\top \rightarrow f(x_1,\ldots,x_n) \dot{\neq} g(y_1,\ldots,y_m))$ falls $f \neq g$ oder $n \neq m$
Zyklizität	$\forall(\top \rightarrow x \dot{\neq} t)$ falls t keine Variable ist und x in t vorkommt

Schreiben Sie ein CHR-Constraint `X neq Y`, das zutrifft, wenn $CT \models$ `X`$\dot{\neq}$`Y`.
Orientieren Sie sich an der Implementierung der Gleichheit an dem RT-Löser aus der Vorlesung.
Um die Disjunktion bei der *Verträglichkeit* und der *Zerlegung* zu implementieren verwenden schreiben Sie ein CHR$^{\vee}$ Constraint `one_neq/2` für zwei gleich lange Argumentlisten, das zutrifft, wenn mindestens ein Listenpaar ungleich ist.
Hinweis: Ein Guard ist bei der Verwendung von `';'` in CHR$^{\vee}$ ist zwingend:
`rule @ Head <=> true | (Goal1 ; Goal2).`
Testziele: (1) `X neq f(X)`, (2) `f(a,X) neq f(X,Y)` und (3) `f(g(X),a) neq f(Y,X)`