Universität Ulm
Fakultät für Informatik
Prof. Dr. Thom Frühwirth

Constraint-Programmierung, WS 02/03
Übungsblatt 7

Abgabe bis Mi, 18.12.02, 10:00

Constraintsystem Terme

Aufgabe 7-1

Die Addition zweier Zahlen in successor-Notation (z.B. 3 als `s(s(s(0)))`) lässt sich durch folgendes Prolog-Prädikat `add` bewerkstelligen:

```
add(0,X,X).
add(s(X),Y,s(Z)):-add(X,Y,Z).
```

Schreiben Sie einen Constraintlöser in CHR, der möglichst viele Vereinfachungen von einzelnen `add`-Constraints machen kann. Zum Beispiel folgt aus der ersten Prolog-Klausel:

```
add(0,Y,Z) <=> Y=Z.
add(X,Y,Y) <=> X=0.
```

Aufgabe 7-2

abgeben, falls Familienname des Abgebenden mit einem der Buchstaben A bis L beginnt

Gegeben sei eine Constrainttheorie CT, die die rein syntaktische Ungleichheit $\dot{\neq}$ zwischen zwei Termen in 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 mittels CHR ein Constraint `ungleich(X,Y)`, das zutrifft, wenn $CT \models \mathtt{X} \dot{\neq} \mathtt{Y}$.
Hinweis: Orientieren Sie sich an der Implementierung der Gleichheit aus der Vorlesung. Um *Verträglichkeit* und *Zerlegung* als Simplifikationsregel zu implementieren, soll folgendes CHR-Constraint definiert werden:
`ungleich_mind_eins(List1,List2)` stellt sicher, daß es mindestens ein i gibt, so daß die i-ten Elemente aus `List1` und `List2` das Constraint `ungleich` erfüllen. Es genügt, wenn die Vereinfachung eines Constraints `ungleich_mind_eins(List1,List2)` darin besteht, alle identischen Paare von Elementen aus den beiden Listen zu entfernen und den Spezialfall von einelementigen Listen zu behandeln.
Beispiel: Das Constraint `ungleich_mind_eins([Z,X,a,c],[Z,Y,b,c])` wird zu dem Constraint `ungleich_mind_eins([X,a],[Y,b])` vereinfacht.
`ungleich_mind_eins([X],[a])` wird zu `ungleich(X,a)` vereinfacht.

Aufgabe 7-3 abgeben, falls Familienname mit einem der Buchstaben M bis Z beginnt

Implementieren Sie die Clark'sche Gleichheitstheorie *CET* möglichst "naturgetreu", d.h. mittels ihrer Axiome als Propagierungsregeln, soweit dies möglich ist. Verwenden Sie folgenden Programmanfang:

```
:- use_module(library(chr)).

handler cet.

option(already_in_store, on).

constraints eq/2.

operator(700,xfx,eq).

% Reflexivitaet
X eq X <=> true.

% Symmetrie
X eq Y ==> Y eq X.
```

`f(X) eq Y` bedeutet z.B. $f(X) \doteq Y$. Die option `already_in_store` bewirkt ein besseres Terminierungsverhalten der Symmetrie-Regel, lesen Sie die Option im CHR-Manual nach.

Anwendungsbeispiele für die nötigen Hilfsprädikate zur Termzerlegung finden Sie in den CHR Online Constraintlösern *Rational trees,...* und *Prolog term manipulation...* und im Sicstus bzw. Yap Prolog Manual.

Gehen Sie schrittweise, Regel für Regel vor. Schreiben Sie Regeln, die unmittelbar zum Widerspruch (`false`) führen, aber dann als erstes im Programmtext.

Testen Sie ihr Programm mit den Unifikationsbeispielen von Übungsblatt 1, Aufgabe 1-1.

Was passiert für eine Anfrage wie `X eq f(Y), Y eq f(X)`? Lösen Sie das Problem, indem Sie eine oder mehrere Regeln für Variablen-Substitution einführen.