

Programmiersprachen II – Wintersemester 2011/2012
Übungsblatt 3
Besprechung der Aufgaben in der nächsten Übung am 26.01.2012.

1 Abgabe

Das Bearbeiten der Aufgaben ist für die Vorlesung Programmiersprachen 2 *nicht* verpflichtend. Eine freiwillige Abgabe ist möglich unter: John.Witulski@uni-duesseldorf.de. Termin für die Besprechung ist Donnerstag der 26.01.2012 14:30 in 25.12.02.55.

2 Aufgaben

Aufgabe 1 (Abstrakte Interpretation einer imperativen Programmiersprache)

Sie haben in Aufgabenblatt 2 einen Interpreter für eine Imperative Sprache erweitert. Nutzen Sie für den weiteren Verlauf dieser Aufgabe diesen Interpreter.

Nehmen Sie folgende Abstrakte Domäne an, mit der die Menge der ganzen Zahlen \mathbb{Z} abstrahiert werden soll: $D_a = \{top, neg, neg_{even}, neg_{odd}, pos, pos_{even}, pos_{odd}, 0\}$. Die abstrakten Elemente $neg_{even}, neg_{odd}, pos_{even}, pos_{odd}$ können genutzt werden, um exaktere Ergebnisse für die Operatoren = und ** zu bekommen.

Hinweis: Sie benötigen keine Abstraktionsmenge, bzw. -Funktion für den Datentyp Bool.

- Implementieren Sie neben den vorhandenen Operatoren folgende Operatoren hinzu: \geq , \leq , or und and.
- Implementieren Sie einen abstrakten Interpreter für den konkreten Interpreter mit der Domäne D_a .

Aufgabe 2 (Co-Routinen)

Betrachten Sie folgendes Programm:

```
app([], R, R).  
app([H|T], X, [H|R]) :- app(T, X, R).  
dappend(A, B, C, R) :- app(A, B, R1), app(R1, C, R).
```

- Wenn das Programm rückwärts ausgeführt wird, d.h. zum Beispiel `dappend(A,B,C,[a,b,c])` laufen Sie nach einer gewissen Anzahl von Lösungen in eine Endlosschleife. Warum passiert das?
- Schreiben Sie `app/3` durch Verwenden von Co-Routinen so um, dass das Programm korrekt arbeitet.

Aufgabe 3 (Co-Routinen 2)

Einführung: Machen Sie sich mit den eingebauten Prädikaten `ground/1` und `var/1` vertraut. Erinnern Sie sich an die 4. Aufgabe im ersten Blatt 0 an das Prädikat `length/2`. Der Aufruf `length(_, X)` kann als ein Generator für Zahlen benutzt werden $X = 0..Maxint$.

Aufgabe: Betrachten Sie folgende Abfrage (Die Ausgabe an ihrer Konsole kann je nach Prolog-System von dieser abweichen):

```
?- 1 is A + 1.  
! Instantiation error in argument 2 of is/2  
! goal: 1 is _114+1
```

Für das Prädikat `is/2` müssen die Parameter auf der rechten Seite *ground* sein, also einen Wert haben.

- a) Schreiben Sie zwei Prädikate `lplus/3` und `lminus/3`, wobei `lplus(A,B,C)` für $A+B=C$ und `lminus(A,B,C)` für $A-B=C$ stehen. Hierbei darf ein beliebiger Parameter eine Variabel sein. Die anderen beiden müssen einen Wert haben.

Beispiel:

```
?- lplus(X,2,3).
```

```
X = 1 ?
```

```
yes
```

```
?- lminus(5,X,2).
```

```
X = 3 ?
```

```
yes
```

- b) Als nächstes soll es möglich sein, die ersten beiden Parameter als Variablen zu übergeben. Es sollen dann Lösungsvorschläge zu den Variabelbelegungen zurückgegeben werden. Achten Sie darauf, dass keine Lösung doppelt vorkommt.

Beispiel (Ihre Ausgabe darf von dieser abweichen, solange sie korrekte Lösungen sind):

```
?- lplus(A,B,3).
```

```
A = 3,
```

```
B = 0 ? ;
```

```
A = 2,
```

```
B = 1 ? ;
```

```
A = 1,
```

```
B = 2 ? ;
```

```
A = 0,
```

```
B = 3 ? ;
```

```
A = -1,
```

```
B = 4 ?
```

```
yes
```

- c) Es soll nun eine beliebige Anzahl an Lösungen gefunden werden. Schreiben Sie dazu das Prädikat `resultset/3`, welches als ersten Parameter einen Aufruf wie in b) erhält, als zweiten Parameter die Anzahl der Lösungen und als dritten Parameter eine Liste mit möglichen Variabelbelegungen als Tupel.

Beispiel (Ihre Ausgabe darf von dieser abweichen, solange sie korrekte Lösungen sind):

```
?- resultset(lplus(_A,_B,3),10,Results).
```

```
Results = [(3,0),(2,1),(1,2),(0,3),(-1,4),(-2,5),(-3,6),(-4,7),(-5,8),(-6,9)] ?
```

```
yes
```

Hinweis: Nutzen Sie das Prädikat `call/N` um die Prädikate `lplus/3` und `lminus/3` aufzurufen, wobei $N \geq 1$. `call` ist ein Metaprädikat in Prolog und ruft eine gegebene Klausel auf.

Beispiel:

```
?- call(atom('wasserstoff')).
```

```
yes
```

Es kann auch folgendermassen verwendet werden:

```
?- call(is,A,1+2).
```

```
A = 3 ?
```

```
yes
```