Aufgabe 4-1

Seien σ, τ, ρ Basistypen. Leiten Sie im einfach getypten λ-Kalkül den Typ von

$$\lambda\, x : (\sigma \to \tau \to \rho).\ \lambda\, y : (\sigma \to \tau).\ \lambda\, z : \sigma.\ x(z)(y(z))$$

her.

Aufgabe 4-2

In dieser Aufgabe sollen Mengen in PVS durch ihre charakteristischen Prädikate formalisiert werden. Der folgende Ausschnitt aus der Theorie `Sets` formalisiert die Mengenzugehörigkeit und die Gleichheit auf Mengen:

```
Sets : THEORY                                                          1
BEGIN
  X   : TYPE
  SET : TYPE = pred[X]

  member(x:X,S:SET) : bool = S(x);
  ==(S:SET,T:SET)   : bool = FORALL x: member(x,S) IFF member(x,T)
END Sets
```

Für Tippfaule ist die Datei `Sets.pvs` als Grundgerüst für diese Aufgabe auf der WWW-Vorlesungsseite hinterlegt.

Hinweis: Zum Auffalten von Definitionen wird der Beweiser-Befehl `(EXPAND)` verwendet, siehe auch die entsprechenden Abschnitte in der ausgeteilten *Kurzeinführung in PVS*.

a) Definieren Sie die leere Menge `emptyset` und die Menge `fullset`, die alle Elemente enthält, in PVS. Zeigen Sie:

```
empty_no_members : LEMMA  NOT member(x, emptyset)                      2
fullset_member   : LEMMA  member(x, fullset)
```

b) Definieren Sie Operatoren `add` und `remove`, die ein Element einer Menge hinzufügen bzw. es aus ihr entfernen. Zeigen Sie:

```
member_add        : LEMMA  member(x, add(x, S))                        3
add_idempotent    : LEMMA  add(x, add(x,S)) == add(x,S)
member_remove     : LEMMA  NOT member(x,remove(x,S))
```

c) Definieren Sie ein Prädikat `subset?` für die Teilmengenbeziehung zwischen Mengen. Zeigen Sie:

```
subset_emptyset      : LEMMA  subset?(emptyset, S)                    4
subset_fullset       : LEMMA  subset?(S, fullset)
subset_antisymmetric : LEMMA  subset?(S, T) AND subset?(T, S) IMPLIES S == T
```

d) Definieren Sie Operatoren für Schnitt und Vereinigung von Mengen. Zeigen Sie:

```
union_commutative : LEMMA  union(S, T) == union(T, S)                  5
union_subset      : LEMMA  subset?(S, T) IMPLIES union(S, T) == T
distribute_union_intersection : LEMMA
  union(S, intersection(T, R)) == intersection(union(S, T), union(S, R))
```

e) Definieren Sie Operatoren für das Komplement einer Menge und die Differenz zweier Mengen. Zeigen Sie:

```
complement_emptyset     : LEMMA complement(emptyset) == fullset        6
demorgan : LEMMA
  complement(intersection(S,T)) == union(complement(S),complement(T))
difference_fullset      : LEMMA difference(S, fullset) == emptyset
difference_intersection : LEMMA difference(S,T) == intersection(S,complement(T))
```

Aufgabe 4-3

In der Vorlesung ist die Definition der Gleichheit in PL2 angegeben worden. Weisen Sie mit PVS Reflexivität, Symmetrie, Transitivität und Substitutivität dieser Gleichheit nach!

Hinweise:

- Unter Substitutivität versteht man, dass Terme an beliebiger Stelle durch gleiche ersetzt werden können. Wir beschränken uns hier auf das Ersetzen innerhalb von Prädikaten:

$$x = y \Rightarrow (P(x) \Leftrightarrow P(y))$$

- Verwenden Sie für Ihre Gleichheit in PVS das Symbol `==`.

Aufgabe 4-4

In dieser Aufgabe soll in PVS das Konzept der transitiven Hülle einer binären Relation definiert werden.

Den Typ einer binären Relation über einem Basistyp `T` definieren wir in PVS als:

```
T      : TYPE
BinRel : TYPE = [T,T -> bool]
```

a) Definieren Sie ein Prädikat `<= : [BinRel,BinRel -> bool]`, so dass `R <= S` genau dann gilt, wenn `S` die Relation `R` umfasst ist.

b) Definieren Sie einen Operator `smallest : [pred[BinRel] -> BinRel]`, so dass `smallest(P)` die kleinste Relation (bzgl. `<=`) liefert, die `P` erfüllt.

c) Definieren Sie ein Prädikat `transitive? : [BinRel -> bool]`, so dass `transitive?(R)` genau dann gilt, wenn `R` transitiv ist.

d) Definieren Sie einen Operator `transclosure : [BinRel -> BinRel]`, so dass `transclosure(R)` die transitive Hülle der Relation `R` liefert.

Hinweis: Die transitive Hülle einer Relation `R` ist die *kleinste* Relation, die *transitiv* ist und `R` *umfasst*.

e) Zeigen Sie, dass ihre Formalisierung der transitiven Hülle tatsächlich die gewünschten Eigenschaften besitzt, nämlich (*i*) transitiv ist, (*ii*) `R` umfasst und (*iii*) die kleinste solche Relation ist.