Aufgaben zur Vorlesung "*Grundlagen der Informatik*"

Aufgabe 1:

(a) Es seien l_1 und l_2 Listen von ganzen Zahlen. Geben Sie eine SetlX-Prozedur `both()` an, so dass der Aufruf `both`(l_1, l_2) die Menge aller der Zahlen berechnet, die sowohl in l_1 als auch in l_2 auftreten. (4 Punkte)

(b) Eine Tripel $\langle x, y, z\rangle \in \mathbb{N}^3$ ist ein *pythagoräisches Tripel* falls

$$x^2 + y^2 = z^2$$

gilt. Beispielsweise ist $\langle 3, 4, 5\rangle$ ein pythagoräisches Tripel, denn es gilt $3^2 + 4^2 = 5^2$. Geben Sie eine SetlX-Prozedur `pythagoras()` an, so dass der Aufruf `pythagoras`(n) die Menge aller der pythagoräischen Tripel $\langle x, y, z\rangle$ berechnet, für die x, y und z alle kleiner als n sind. (6 Punkte)

(c) Schreiben Sie eine Prozedur `subsets()`, so dass der Aufruf

$$\mathit{subsets}(n, k)$$

für zwei natürliche Zahlen n und k die Menge aller der Teilmengen der Menge $\{1, \cdots, n\}$ berechnet, die genau k Elemente enthalten. (8 Punkte)

Aufgabe 2: Wir definieren eine Abstands-Relation A als eine Menge von Paaren der Form $\big\langle\langle x, y\rangle, d\big\rangle$ wobei gilt:

1. $\langle x, y\rangle \in P \times P$.
2. $d \in \mathbb{N}$ und es gilt $d \geq 1$.

Wir interpretieren eine Abstands-Relation A wie folgt: Ist $\big\langle\langle x, y\rangle, d\big\rangle \in A$, so gibt es eine direkte Verbindung zwischen den Punkten x und y und diese Verbindung hat die Länge d. Die Komposition $A_1 \circ A_2$ zweier Abstands-Relationen sei so definiert, dass das Paar $\big\langle\langle x, z\rangle, d\big\rangle \in A$ genau dann ein Element der Komposition $A_1 \circ A_2$ ist, wenn es einen Punkt $y \in P$ gibt, so dass es einerseits in A_1 eine direkte Verbindung von x nach y einer Länge d_1 und andererseits in A_2 eine direkte Verbindung von y nach z der Länge d_2 gibt, so dass $d = d_1 + d_2$ ist.

(a) Implementieren Sie eine SetlX-Prozedur `compose()`, so dass der Aufruf `compose`(A_1, A_2) für zwei Abstands-Relationen A_1 und A_2 die Komposition $A_1 \circ A_2$ berechnet. (8 Punkte)

(b) Implementieren Sie eine Prozedur `closure`, so dass der Aufruf `closure`(A) für eine Abstands-Relation A die Menge aller Paare $\big\langle\langle x_1, x_n\rangle, d\big\rangle$ berechnet, so dass es eine endliche Folge $x_1, x_2, x_3, \cdots, x_n$ von Punkten aus P gibt, so dass einerseits

$$\big\langle\langle x_i, x_{i+1}\rangle, d_i\big\rangle \in A$$

und andererseits $d = d_1 + d_2 + \cdots + d_{n-1}$ gilt. (10 Punkte)

Hinweis: Sie dürfen voraussetzen, dass es nur endlich viele Pfade gibt.

Aufgabe 3: Implementieren Sie eine SetlX-Prozedur `toLists`(), so dass der Aufruf `toLists`(M) für eine Menge M die Menge aller Listen berechnet, die jedes Element der Menge M genau einmal enthalten. Beispielsweise soll der Aufruf

`toLists`($\{1, 2, 3\}$)

als Ergebnis die Menge

$$\{ [1, 2, 3],\ [1, 3, 2],\ [2, 1, 3],\ [2, 3, 1],\ [3, 1, 2],\ [3, 2, 1] \}$$

berechnen. (8 Punkte)

Hinweis: Implementieren Sie die Prozedur rekursiv.

Aufgabe 4:

(a) Formalisieren Sie die Schluß-Regel, die in dem folgenden Argument verwendet wird.

> *Wenn der linke Motor ausfällt oder wenn der rechte Motor ausfällt, stürzt das Flugzeug ab. Das Flugzeug stürzt ab. Der linke Motor fällt nicht aus. Also ist der rechte Motor ausgefallen.*

Hinweis: Die verwendete Schluß-Regel hat drei Prämissen. (4 Punkte)

(b) Überprüfen Sie, ob diese Schluß-Regel korrekt ist. (6 Punkte)

Aufgabe 5:

(a) Betrachten Sie die folgende Schluß–Regel:

$$\frac{q \wedge p \qquad r \vee \neg q}{r \wedge p}$$

Geben Sie eine aussagenlogische Formel f an, die genau dann eine Tautologie ist, wenn die obige Schluß–Regel korrekt ist. (2 Punkte)

(b) Berechnen Sie die konjunktive Normalform der in (a) berechneten Formel. (8 Punkte)

Aufgabe 6: Der Begriff eines *arithmetischen* SetlX-*Ausdrucks* werde wie folgt induktiv definiert:

1. Jede String ist ein *arithmetischer* SetlX-*Ausdruck*. Ein String stellt eine *Variable* dar.
2. Sind s und t *arithmetische* SetlX-*Ausdrücke*, so sind auch
   (a) $[s, \text{“}\texttt{+}\text{”}, t]$,
   (b) $[s, \text{“}\texttt{-}\text{”}, t]$,
   (c) $[s, \text{“}\texttt{*}\text{”}, t]$,
   (d) $[s, \text{“}\texttt{/}\text{”}, t]$

   aritmetische SetlX-*Ausdrücke*.

(a) Schreiben sie eine Prozedur `allVars`(s), die für einen gegebenen arithmetischen SetlX-Ausdruck die Menge aller Variablen berechnet, die in s auftreten.
(8 Punkte)

(b) Schreiben sie eine Prozedur `countVars`(s, x), die für einen gegebenen arithmetischen SetlX-Ausdruck s und eine Variable x zählt, wie oft die Variable x in s auftritt.
(8 Punkte)

(c) Schreiben sie eine Prozedur `singleVars`(s), die für einen gegebenen arithmetischen SetlX-Ausdruck die Menge aller Variablen berechnet, die in s genau einmal auftreten.
(4 Punkte)

Aufgabe 7: Die Menge M sei wie folgt definiert:

$$M := \big\{\{t, q, r\}, \{t, p, \neg r\}, \{\neg t, p, r\}, \{t, \neg q\}, \{\neg t, p\}, \{q, \neg p\}, \{\neg t, \neg p\}\big\}.$$

Zeigen Sie durch mehrfache Anwendung der Schnitt-Regel, dass $M \vdash \{\}$ gilt. (8 Punkte)

Aufgabe 8: Die Menge M sei wie folgt definiert:

$$\begin{aligned} M := \{ \ & \{p, q\}, \{\neg p, r, \neg t\}, \{r, s, p\}, \{\neg r, q, \neg p\}, \{r, \neg s, p\}, \\ & \{\neg p, \neg q, s\}, \{p, \neg q, s\}, \{\neg r, \neg s\}, \{\neg p, \neg s\}, \{q, \neg p, t\}\} \end{aligned}$$

Zeigen Sie mit dem Verfahren von Davis-Putnam, dass M nicht lösbar ist. (12 Punkte)

Aufgabe 9: Geben Sie eine prädikatenlogische Struktur $\mathcal{S}$ an, in der die Formel

$$(\forall x \colon \exists y \colon p(x, y)) \rightarrow \exists y \colon \forall x \colon p(x, y)$$

nicht gilt. Die prädikatenlogische Struktur soll wie im Skript mit Hilfe von SetlX beschrieben werden. (10 Punkte)

Aufgabe 10: Wandeln Sie die folgende Formel in prädikatenlogische Klausel-Normalform um und geben Sie das Ergebnis in Mengen-Schreibweise an.

$$\Big(\forall x \colon \neg\exists y \colon \big(q(x) \rightarrow p(x, y)\big)\Big) \rightarrow \big(\forall z \colon q(z)\big)$$

(8 Punkte)

Aufgabe 11: Implementieren Sie ein *Prolog*-Prädikat "`is_contained`" mit der Typ-Spezifikation

$$\texttt{is\_contained}(+\mathit{List}(\mathit{Number}), +\mathit{List}(\mathit{Number})).$$

Ein Aufruf von `is_contained` soll die Form

$$\texttt{is\_contained}(l_1, l_2)$$

haben. Dabei sind l_1 und l_2 Listen von Zahlen. Der Aufruf soll genau dann erfolgreich sein, wenn alle Elemente von l_1 in der Liste l_2 enthalten sind. (8 Punkte)

Hinweis: Sie dürfen das eingebaute Prädikat `member` benutzen.

Aufgabe 12: Implementieren Sie ein *Prolog*-Prädikat "`all_lists`" mit der Typ-Spezifikation

$$\texttt{all\_lists}(+\mathit{Number}, +\mathit{Number}, -\mathit{List}(\mathit{Number})).$$

Der Aufruf

$$\mathit{all\_lists}(N, K, L)$$

soll für zwei natürliche Zahlen N und K eine Liste L erzeugen, deren Elemente aus der Menge $\{1, \cdots, N\}$ stammen und die die Länge K hat. Die Liste darf kein Element mehrfach enthalten. Außerdem soll das Prädikat beim Backtracking alle möglichen Listen mit der oben beschriebenen Eigenschaft erzeugen. (8 Punkte)