Technische Universität München
Fakultät für Informatik
Prof. Dr. H. Seidl

SoS 2006
Übungsblatt 12
19. Juli 2006

Übungen zu Einführung in die Informatik II

Aufgabe 1 Datalog

a) Welche der nachfolgenden Datalog-Programme sind korrekt? Begründen Sie Ihre Antwort.

(i)
```
p(X) :- not (q(X)).
q(a).
```

(ii)
```
p(a) :- p(a), not (p(a)).
```

(iii)
```
mem(a). mem(b). mem(c). mem(d). mem(e).
rel(a,b). rel(b,c). rel(c,d). rel(c,e).
tc(X,Y) :- rel(X,Y).
tc(X,Y) :- tc(X,Z),rel(Z,Y).
p(X) :- mem(X), not (tc(b,X)).
```

b) Betrachten Sie das Datalog-Programm iii) als gegeben. Geben Sie die Ergebnisse zu den folgenden Anfragen an:

(i)
```
?- mem(c).
```

(ii)
```
?- rel(c,X).
```

(iii)
```
?- tc(X,d)
```

(iv)
```
?- p(X).
```

c) Gegeben sei folgendes Datalog-Programm:

```
person(angela).     person(georg).     person(franz).     person(peter).
mag(angela,georg).  mag(georg,angela). mag(franz,angela). mag(peter,franz).
```

Definieren Sie ein Prädikat

(i) `gegenseitig`, dass von allen Paaren von Leuten erfüllt wird, die sich gegenseitig mögen.

(ii) `q`, dass von allen Personen erfüllt wird, die sowohl von Georg als auch von Angela gemocht werden.

(iii) `p`, dass von allen Personen erfüllt wird, die Angela oder Franz mögen.

d) Ergänzen Sie die Definition des Prädikates `mag`, so dass

(i) jeder sich selbst mag;

(ii) jeder diejenigen mag, die von jemandem gemocht werden, den er mag.

Aufgabe 2 Breitensuche in OCaml

Neben der bereits bekannten Tiefensuche ist die *Breitensuche* ein weiterer Algorithmus, um systematisch alle Knoten (und Kanten) eines Graphen $G = (V, E)$ zu besuchen.

Die Spezifikation des Verfahrens ist durch den folgenden Pseudocode gegeben. Dabei seien $V = \{v_1, \ldots, v_n\}$, v_1 der Startknoten, und bezeichne $\Gamma(v)$ die Menge der Nachbarknoten eines Knoten v.

```
Queue Q;
Q.push (v1);

while (! Q.empty ()) do
   u = Q.pop ();

   foreach v in Gamma (u) do
      if (! visited (v)) then
         Q.push (v);
      endif
   done
done
```

a) Im obigen Pseudocode wird eine Warteschlange (Queue bzw. FIFO) verwendet. Implementieren Sie hierzu die entsprechenden auf Listen arbeitenden OCaml-Funktionen `fifo_push`, `fifo_pop`, `fifo_empty` und `fifo_init`, die die Verwendung einer Liste als FIFO ermöglichen.

b) Bei der Implementierung eines FIFOs mit einer einzelnen OCaml-Liste ist mindestens eine der obigen Operationen ineffizient. Beheben sie diesen Mißstand durch die eine alternative Implementierung der obigen Funktionen, die auf einem Paar von Listen (l_1, l_2) arbeiteten.

   Hierbei soll die erste Liste l_1 zum Einfügen von neuen Elementen (`push`), und l_2 zum Entfernen (`pop`) des vorderstens Elements verwendet werden. Immer dann, wenn l_2 beim Aufruf einer der Funktion `pop` leer ist, und l_1 noch Elemente enthält, soll dafür gesorgt werden (wie?), dass sich wieder Elemente in l_1 befinden.

c) Implementieren Sie eine Funktion `bfs: int list array -> int-> int list`, die die Knoten eines Graphen ausgehend von einem Startknoten s in einer Breitensuche besucht und eine Liste zurück gibt, die die Knoten in der Reihenfolge des Besuchs enthält.

d) Definieren Sie zunächst eine Signatur `Fifo`, die die Schnittstelle für den Zugriff auf eine Warteschlange definiert und und damit von der zugrundeliegenden Implementierung abstrahiert.

   Integrieren Sie nun die beiden oben angefertigten FIFO-Implementierungen durch die Definition der Strukturen `ListFifo` bzw. `ListPairFifo`.

e) Als letzter Schritt ist nun ein Funktor `BreadthFirstSearch` zu implementieren, der mit einer FIFO-Implementierung parametrisiert werden kann, und eine leicht modifizierte Variante der Funktion `bfs` enthält.