SE3

Aufgabenblatt 7

Softwareentwicklung 3: Logikprogrammierung - WS 2015/2016 - W. Menzel

Listenverarbeitung

Gesamtpunktzahl: 30 Abgabe der Lösungen bis zum 7.12.2015

Aufgabe 1: Deduktive Datenbanken (6) 10 Punkte
maximale Bearbeitungszeit: 40 Minuten

Wir betrachten wieder das in der Datei `dateiverzeichnis.pl` gegebene Datenbankschema.

1. Definieren Sie ein Prädikat, das für ein durch seinen Index identifiziertes *Verzeichnis*, den Zugriffspfad als Liste von Verzeichnisnamen ermittelt.

2. Definieren Sie ein Prädikat, das für eine gegebene *Datei*, den Zugriffspfad ermittelt. Die Datei soll durch ihren Namen und den Index des zugehörigen Verzeichnisses identifiziert werden.

3. Definieren Sie ein Prädikat, das eine geordnete Liste der n zuletzt modifizierten Dateien ermittelt. Die Liste soll sowohl die Datei-Identifikatoren, als auch die zu ihnen führenden Zugriffspfade enthalten. Beachten Sie dabei, dass n größer als die Anzahl der Dateien in der Datenbank sein kann.

   Entwerfen Sie für dieses Problem zuerst eine Lösungsstrategie und leiten sie daraus Anforderungen an eine geeignete Repräsentation für die Zwischenergebnisse ab. Setzen Sie dann die Teilschritte Ihres Entwurfs in entsprechende Prädikatsdefinitionen um.

   Diskutieren Sie Ihre Lösung im Hinblick auf ihren Rechenzeitbedarf und eventuelle Möglichkeiten zu seiner Reduzierung.

   Hinweis: Mit den eingebauten Prädikaten `sort/2` bzw. `msort/2` können Sie auch Listen von komplexen Termen sortieren.

Aufgabe 2: verzweigende rekursive Strukturen 11 Punkte
maximale Bearbeitungszeit: 70 Minuten

Im Skript zur Vorlesung wird auf Seite 117 gezeigt, wie man einen Binärbaum konstruiert. Hier soll die Implementation so erweitert werden, dass man den Baum für den Zugriff auf ein Wörterbuch verwenden kann.

1. Überlegen Sie sich, welche Form der durch das Prädikat `list2tree/2` erzeugte Baum für unterschiedliche Eingabelisten annimmt. Wie müsste die Eingabeliste sortiert sein, damit der entstehende Baum balanciert ist?

2. Erweitern Sie die Repräsentation für einen Baumknoten so, dass an den einzelnen Knoten zusätzliche Informationen zu den jeweiligen Suchbegriffen abgespeichert werden können. Modifizieren Sie das Prädikat `list2tree/2`, damit es mit der neuen Repräsentation arbeiten kann.

3. Definieren Sie ein Prädikat, mit dem Sie auf die Information zu den Suchbegriffen zugreifen können

4. Definieren Sie ein Prädikat, mit dem Sie einen balancierten Suchbaum aufbauen können.

Aufgabe 4: Zeichenkettenvergleich 9 Punkte

maximale Bearbeitungszeit: 60 Minuten

Für die Textverarbeitung und die fehlertolerante Suche in Datenbanken wird ein Prädikat benötigt, das die (Un-)Ähnlichkeit von zwei gegebenen Zeichenfolgen ermittelt, um potenzielle Korrekturkandidaten bzw. orthografisch leicht abweichende Datenbankeintragungen zu finden. Die Ähnlichkeit kann auf der Basis einer paarweisen Zuordnung von übereinstimmenden bzw. nicht übereinstimmenden Zeichen in den beiden Zeichenfolgen ermittelt werden. Implementieren Sie die folgenden Prädikate, um derartige paarweise Zuordnungen zu berechnen. Benutzen Sie Listen für die Repräsentation der Zeichenfolgen. Für die Umwandlung zwischen Atomen in Listen können Sie das Prädikat `atom_chars/2` verwenden.

1. Definieren Sie ein Prädikat, das für zwei *gleich lange* Listen von Zeichen die Anzahl der *nicht* übereinstimmenden Zeichenpaare ermittelt (HAMMING-Distanz).

2. Erweitern Sie das Prädikat aus Teilaufgabe 1 so, dass auch die HAMMING-Distanz für Listen unterschiedlicher Länge berechnet werden kann. Dabei sollen die überzähligen Zeichen als nicht übereinstimmende Paare in das Distanzmaß eingehen. Fügen Sie dazu zwei neue Klauseln zu Ihrer Definition hinzu, die die beiden Fälle abdecken, wenn eine der beiden Listen leer ist, die andere jedoch nicht.

3. Erweitern Sie das Prädikat aus Teilaufgabe 2 so, dass es neben der HAMMING-Distanz auch die paarweise Zuordnung der Zeichen ausgibt, für die beiden Zeichenketten `wand` und `wunder` z.B. in der Form

   ```
   [[w,w],[a,u],[n,n],[d,d],[*,e],[*,r]]
   ```

4. Die Annahme, dass überzählige Zeichen nur am Ende einer Zuordnung auftreten können, ist nicht realitätsgerecht. Erweitern Sie daher Ihre Prädikatsdefinition aus Teilaufgabe 3 durch zwei weitere Klauseln, die Einfügungen bzw. Weglassungen von Zeichen an beliebigen Stellen in den Zeichenfolgen zulassen. Beachten Sie dabei, dass sich dadurch alternative Zuordnungen (mit unterschiedlichen Distanzwerten) ergeben. Die minimale Distanz über all diesen unterschiedlichen Zuordnungen wird auch als LEVENSTHEIN-Metrik bezeichnet.

   Untersuchen Sie, ob die Menge der von Ihrem Prädikat ermittelten Zuordnungen endlich ist. Untersuchen Sie auch, ob das Prädikat Doppelergebnisse berechnet.

5. Nur für Interessenten: `alignment/4` sei das in Teilaufgabe 4 entwickelte Prädikat, das alle Zuordnungen zwischen zwei Zeichenfolgen und die zugehörige Distanz ermittelt. Die folgende Implementation zur Berechnung der LEVENSHTEIN-Distanz

   ```
   % levenshtein(+Liste1,+Liste2,?Levenshtein_Distanz)
   levenstein(L1,L2,LDistanz) :-
     findall(Distanz,alignment(L1,L2,Distanz,_),Distanzen),
     min_list(Distanzen,LDistanz).
   ```

   ist leider nicht effizient. Diskutieren Sie die Ursachen für diese Ineffizienz und zeigen Sie Möglichkeiten zu ihrer Beseitigung auf. (4 Punkte)