Übungsaufgaben zum SML-Kurs, SS 2003

Hans Leiß

21. Mai 2003

Abgabe: Mi, 28.5.2003

4 Struktur und Signatur für Suchbäume

Aufgabe 4.1 Sammle die Fehlermeldungen von SML, die bei den folgenden (oder früheren) Aufgaben auftreten und unverständlich sind, damit wir sie in der Übungsstunde besprechen können. Ebenso mit anderen auftretenden Fragen! (Versuche im Emacs, mit `C-x ‘` oder über `Process/goto-next-error` im SML-Menü an die Fehlerstelle zu springen; hilft das?) (4 Punkte)

Wir wollen eine Art von Bäumen definieren, die zur Speicherung und Suche von Informationen geeignet ist, die mit Zeichenreihen angesprochen werden soll. Z.B. werden in einem Adressenverzeichnis zu jedem Namen eine Reihe von Informationen wie Telephonnummer, Wohnort, Geburtsdatum usw. verzeichnet, und diese Informationen werden über den Namen gesucht.

Die erste Idee ist, einfach eine Liste `[(Namen,Daten),...]` anzulegen. Aber dann wird die Suchdauer sehr stark davon abhängen, ob der jeweilige Name eher am Anfang oder eher am Ende der Liste auftritt. Außerdem wiederholt sich in den Namensfeldern Information, wenn z.B. Vor- und Nachnamen angegeben sind, und das kann zu unnötigem Speicherbedarf führen.

Beide Probleme kann man vermeiden, wenn man statt einer Liste eine bessere Datensruktur zur Speicherung verwendet. Man sollte gemeinsame Präfixe der Namen nur einmal darstellen und dann „lexikographisch“ suchen: alle Informationen zu Namen, die mit dem Buchstaben `A` beginnen, sollten in einem anderen Teil der Datenstruktur stehen als die Inforamtionen zu Namen, die mit `B` beginnen, usw. Spaltet man entsprechend den Teilbereich, wo die Informationen zu Namen der Form `A ...` stehen, nach dem zweiten Buchstaben der Namen auf usw., so erhalten wir als Datenstruktur einen *Baum, dessen Kanten mit Buchstaben und dessen Blätter und Verzweigungsknoten mit Informationen versehen sind oder sein können.* An einem Knoten steht die Information zu dem Namen, der sich aus der Buchstabenfolge an den Kanten von der Wurzel zu diesem Knoten ergibt.

Wenn man noch Knoten, die nur zu einem einzigen Teilbaum führen, wegläßt und dafür an den Kanten Buchstabenfolgen zuläßt, nennt man diese Datenstruktur einen *Trie* (von „Information re*trie*val“, und weil es eine Art Baum, also *Tree*, ist). Wir implementieren nur die einfache Form, wo an den Kanten einzelne Buchstaben stehen.

Wir wollen uns aber nicht auf den Fall beschränken, wo die Kanten mit Buchstaben beschriftet sind, sondern Objekte eines beliebigen Typs `'a` zulassen. Beispielsweise möchte man Daten unter Kundennummern oder Binärzahlen speichern, und Ziffernfolgen als „Suchwörter“ benutzen.

Aufgabe 4.2 Schreibe eine Signatur

⟨*Trie.sml*⟩≡
```
structure TRIE =
   sig
      type ('a,'b) t         (* Typ der Tries mit Kantenbeschriftung vom Typ 'a
```

```
                              und Informationen vom Typ 'b an den Knoten *)
    val empty : ('a,'b) t  (* Leerer Trie *)
    val insert : ...
    val lookup : ...
  end
```

Die Funktion `insert` dient zum Einfügen einer Information unter einem Namen (bwz. zum Überschreiben der vorhandenen Information), und die Funktion `lookup` dient dazu, aus einem Trie und einer `'a`-Liste (!) – z.B. der `char`-Liste `(explode Name")` – als Antwort die im Trie gespeicherte Information abzulesen.

Man sucht also mit einer `'a`-Liste. Falls keine Information vorliegt, kann die Antwort nicht vom Typ `'b` der Informationen sein. Für solche Zwecke hat SML den eingebauten Datentyp

⟨*option*⟩≡

```
datatype 'b option = SOME of 'b | NONE ;
```

Die Funktion `lookup` sollte daher einen Typ der Form `( ... -> 'b option)` haben. Vervollständige die Signatur durch Angabe der passenden Typen. (4 Punkte)

Aufgabe 4.3 (i) Schreibe eine Struktur

⟨*Trie.sml*⟩+≡

```
structure Trie =
   struct
    ...
   end
```

die eine Definition des Datentyps und der gewünschten Funktionen `empty`, `insert`, `lookup` enthält.

Hinweis: Bei einem Verzweigungsknoten gibt man die direkten Teilbäume am einfachsten in einer Liste von Paaren an, bei denen die erste Komponente die Marke an der Kante vom Verzweigungsknoten zu diesem Teilbaum enthält. Beachte, daß bei Einfügen von Daten zu einem „Suchwort“ die Daten nur an dem Knoten geändert werden, der durch die `'a`-Liste der Elemente des „Suchworts“ von der Wurzel aus erreicht wird. (Falls man neue nicht-Blattknoten anlegen muß, muß man dort die Information `NONE` ablegen!) (8 Punkte)

(ii) Konstruiere mit `empty` und `insert` einen Trie `tr`, indem zu Ziffenfolgen Zeichenreihen gespeichert werden, der zum „Suchwort“ `[1,2,3]` das Datum `"Aachen"` und zu `[1,2,4]` das Datum `"Augsburg"` enthält. Füge ein Datum zum Suchwort `[1,2]` hinzu.

Konstruiere analog mit `insert (explode <String>) ...` einen Trie, wo zu Wörtern Daten abgelegt und gesucht werden können, z.B. zu Vollformen eines Deutschen Worts seine Stammform. (3 Punkte)

Aufgabe 4.4 Vergleiche die von SML gefundene „allgemeinste“ Signatur der Struktur `Trie` mit der Signatur `TRIE`. Erzeuge durch transparente und durch opake Einschränkungen von `Trie` auf die Signatur `TRIE` zwei Strukturen `Trie1` und `Trie2`.

Demonstriere den Unterschied an Beispielen: kann man die Implementierungstypen der Tries erkennen, kann man zwei Tries desselben oder verschieden Pakete vergleichen? (4 Punkte)