Seminar “Implementierung von Stringalgorithmen”

Leiß/Goller, CIS

21.4.2004

1 Implementierung eines Search-Tries

Aufgabe 1.1 Wahlweise in C++ oder in SML soll ein Search-Trie implementiert werden. Ein Search-Trie ist eine Datenstruktur aus Knoten und Übergängen (mit Labeln), in Form eines Baumes, d.h. insbesondere mit einer Wurzel. Die Labels an den Übergängen sind einzelne Buchstaben, und an jedem Knoten soll es für jeden Buchstaben des Alphabets höchstens einen Übergang zu anderen Knoten geben. Gewisse Knoten sind *wesentliche* oder *End*knoten oder *Endzustände* (Die –in Abb. 1 zufällig mit den Blättern zusammenfallenden– Endknoten werden durch Kästen, die übringen durch Kreise dargestellt.)

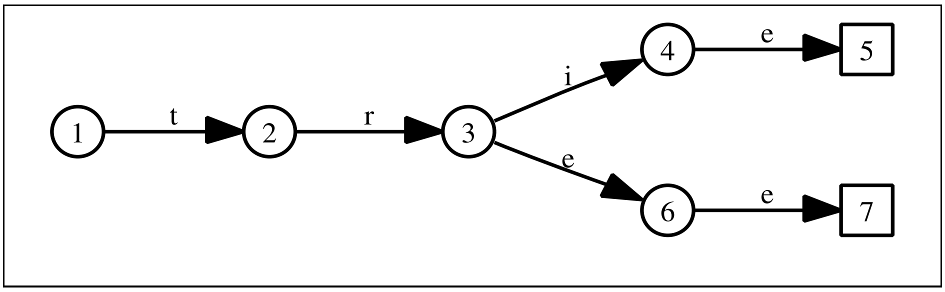

Abbildung 1: Beispiel-Searchtrie. Er enthält die Strings “trie” und “tree”.

Wir sagen, der Trie *enthält* einen String, wenn es einen Pfad von der Wurzel zu einem Endknoten gibt, dessen Labels genau den Buchstaben des Strings (in derselben Reihenfolge) entsprechen.

Für den Zweck dieser Übungsaufgabe soll ein kleines Modul entstehen, das es erlaubt, eine solche Datenstruktur aufzubauen, Strings einzufügen, und zu testen, ob ein bestimmter String enthalten ist.

- In C++ könnte etwa folgendes Interface als Muster dienen:

```
class trie {
public:
  trie();
  ~trie();
  void insert(const std::string &s);
  bool haskey(const std::string &s);
};
```

  Hierfür ist es sicher sinnvoll, Hilfsklassen für die Knoten- und/oder Kanten-Objekte zu erstellen.

- In SML könnte folgende Signatur das Interface beschreiben:

```
signature TRIE =
   sig
       datatype trie = ...
       val empty : trie
       val insert : trie * string -> trie
       val haskey : trie * string -> bool
   end
```

  Vervollständigen Sie die Signatur und Schreiben Sie den Modul als Struktur `structure Trie:TRIE`.