Kapitel 3

Symboltabelle

Symboltabelle: Ziele und Kriterien

Ziele:

- Die Merkmale bzw. Schlüssel aller Symbole festlegen, die nicht durch Endzustände des Automaten bestimmt sind.
- Bezeichner und Konstanten durch Merkmal einheitlicher Länge codieren.
- Aufbewahrung der Bezeichner- und Konstantentexte für die weitere Bearbeitung und für Fehlermeldungen.
- Ankerpunkt, von dem aus verschiedene Vereinbarungen eines Bezeichners erreichbar sind (für semantische Analyse).

Kriterien:

- Anfangs unbekannte Anzahl von Bezeichnern und Konstanten unbeschränkter (!) Länge aufnehmen (Faustregel: pro 10 Zeilen 1 Bezeichner).
- Suche nach Bezeichnertexten wenn möglich mit Aufwand $O(1)$.

Hinweis: Für die lexikalische Analyse wird in der Regel eine Symboltabelle nicht unbedingt benötigt, sie dient als „Gedächtnis“.

ADT Symboltabelle

Gelieferte Operationen:

- `insert(text, key)`
- `find_or_insert(text) : (key,value)`
- `get_text(key) : text`

Benötigte Operationen:

keine

`insert` dient für Voreinträge (z.B. while)

Suchverfahren in Symboltabelle

- Sequentielle Suche (nie sinnvoll)
- Suchbaum (in Fortran 77 nötig)
- Hashen
  - Perfektes Hashen (nur bei vorher bekannter Bezeichnermenge)
  - Verkettetes Hashen (Aufwand?)
  - Hashen mit quadratischem Sondieren o.ä. (Aufwand?)

$$index_i := (h(x) + i^2 \cdot g(x)) mod |Tabelle|$$

i: i-te Kollision, h: Hashfunktion, g: optionale Hashfunktion (ggf. $g := 1$)

  - Einträge: Verweise auf Symbole, gleichzeitig deren Merkmal

Achtung: Bestimmte Implementierung der Symboltabelle kann von der Quellsprache erzwungen werden, siehe Fortran

Umstiegspunkt für Symboltabelle

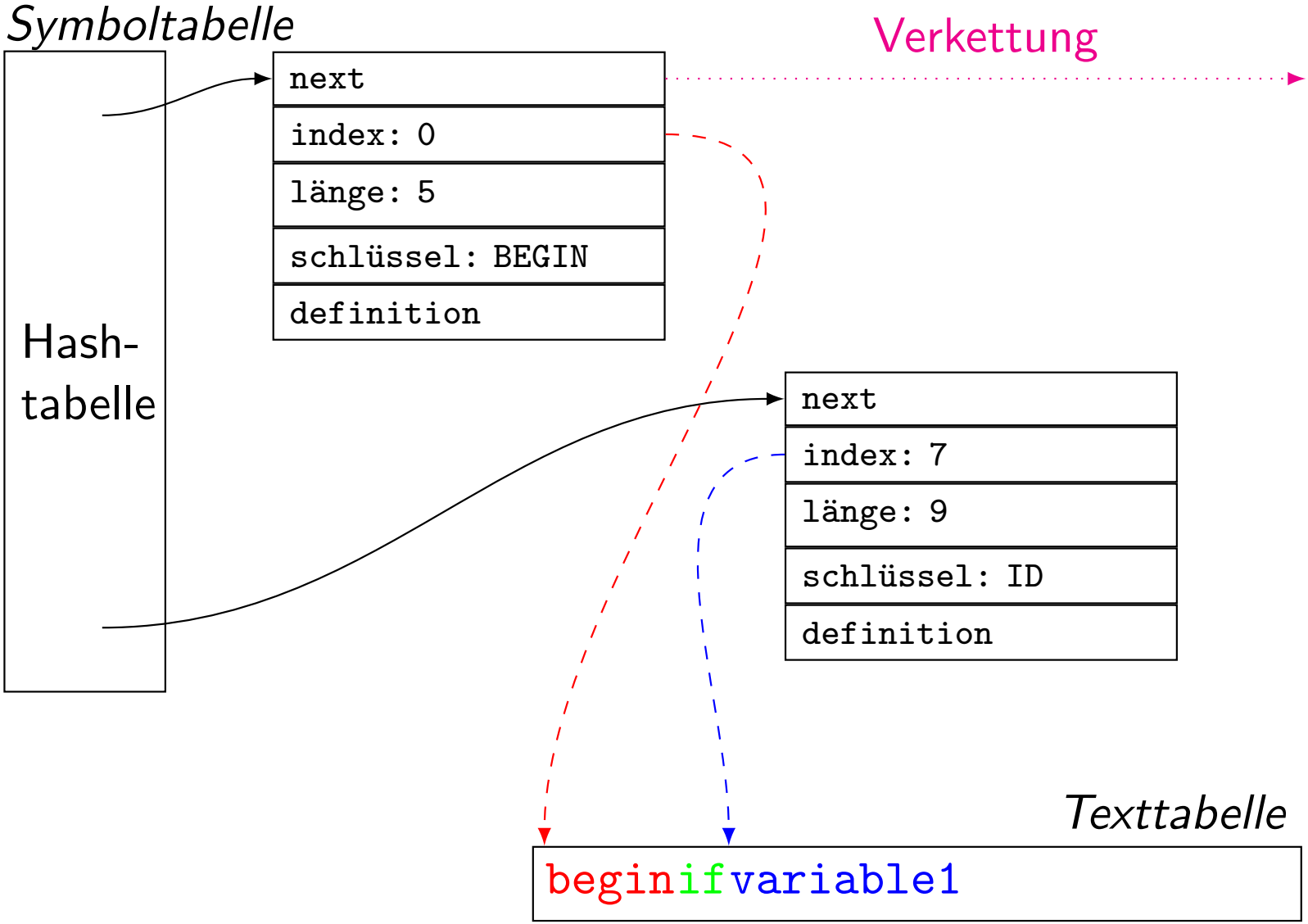
Alternativen:

- Perfektes Hashen: bestes Verfahren wenn anwendbar
- Sondieren (z.B. quadratisch):
  Aufwand: Hashfunktion + # Kollisionen
- verkettetes Hashen: Aufwand: Hashfunktion + Kettenlänge
- Suchbaum: Aufwand Pfadlänge im Baum ($O(\log(|\text{Einträge}|))$)
  in formatiertem FORTRAN sind Leerzeichen erlaubt, ständiges Berechnen von Hashschlüsseln und Suchen in Tabelle ist zu teuer, im Baum ist binäre Suche möglich

Wann welche Technik?

- perfektes Hashen, wenn möglich
- quadratisches Sondieren mit Aufwand $O(1)$, wenn Hashfunktion gleichverteilt und Hashtabelle nur halbvoll; Abhilfe Tabellenverdopplung
- verkettetes Hashen: $O(1)$, wenn Hashfunktion ungefähr gleichverteilt, keine Abhilfe, wenn letzteres nicht erfüllt ist
- Suchbaum: sonst

Organisation der Symboltabelle



ADT Symboltabelleneintrag

```
abstract class SymbolTableEntry is
  next:     SymbolTableEntry; // Verkettung
  index:    Integer;          // Index in Texttabelle
  length:   Integer;          // Laenge in Texttabelle
  key:      Key;              // Art des Eintrags
   definition : DefinitionTableEntry;
     // Momentan gueltige Definition
end SymbolTableEntry;
```

Verweise auf Objekte dieses Typs als Merkmal für Bezeichner usw.

Hashen

- Tabellenlänge: Abstand zu $n \times 256$!
  - Primzahl
  - 2^p (Vermeiden von ganzzahliger Division), p kein Vielfaches von 8!
- Hashverfahren:
  - Hashen mit quadratischem Sondieren, ...
    - verlangt eventuell Tabellenverlängerung
  - verkettetes Hashen
- Faustregel: ca. ein Neueintrag für 10 Zeilen Quelltext
- Hashtabelle nach Ende der lexikalischen Analyse überflüssig, nur Symboleinträge und Texttabelle werden noch benötigt

Hashfunktionen

- Hashfunktion muß schnell berechnet werden können!
- Anfang/Ende vieler Bezeichner gleich (a1, a2, ...)
- $\mathrm{id} = \mathrm{id}_1\mathrm{id}_2 \ldots \mathrm{id}_k$
  - $h(\mathrm{id}) = \mathrm{abs}(\mathrm{id}_1) + \mathrm{abs}(\mathrm{id}_k) + \mathrm{abs}(\mathrm{id}_{(k+1)/2})$
  - $h(\mathrm{id}) = c_1 \times \mathrm{abs}(\mathrm{id}_1) + c_2 \times \mathrm{abs}(\mathrm{id}_k) + c_3 \times \mathrm{abs}(\mathrm{id}_{(k+1)/2})$
    - $c_i = 1, 4, 8$? Spreizung über $0 \ldots 255$, abhängig vom Zeichensatz
  - $h(\mathrm{id}) = \Sigma_{i=1}^{k} c_i \times \mathrm{abs}(id_i)$
  - ...
  - alle Berechnungen modulo Tabellenlänge
- bei Hashfunktionen kommt es auf Gleichverteilung an, nicht auf die aktuelle Rechenmethode
  - z.B. spart wortweises Addieren Zeit (verletzt aber die Typregeln)

Kosten einer Tabellenverdopplung

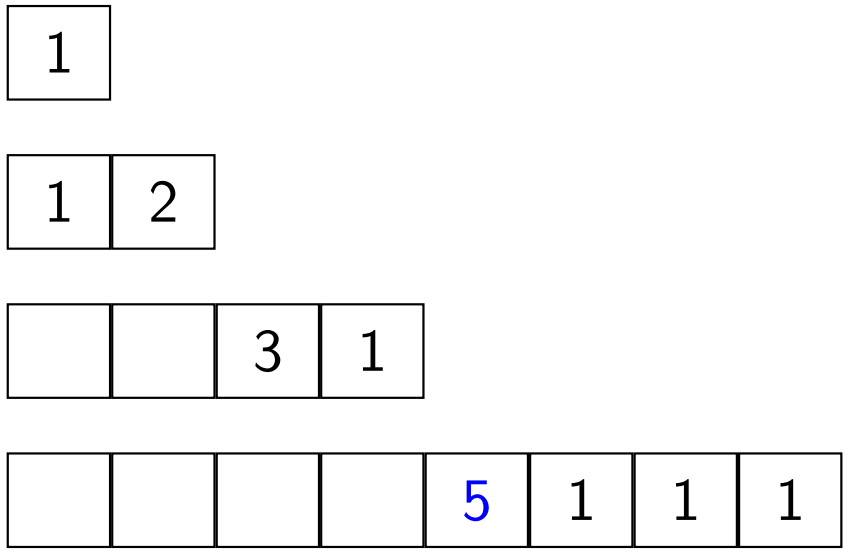
Sei

$$n = 2^k$$

Zugriffszeit für n Elemente:

$n+n/2+n/4+n/8+\ldots < 2n \in O(n)$

amortisiert für ein Element: $O(1)$



Betrachte Knoten 5: 4 Schreibzugriffe für Kopieren + eigentlicher Schreibzugriff

Achtung: Der Index kann nicht das Merkmal eines Symbols sein, da er sich offenbar ändern kann.