Grundlagen der Programmierung (Vorlesung 16)

Ralf Möller, FH-Wedel

- Vorige Vorlesung
  - Blöcke, Funktionen, Prozeduren
  - Auswertestrategien, Rekursion
- Inhalt dieser Vorlesung
  - (Asymptotische) Komplexität von Algorithmen
- Lernziele
  - Grundlagen der Analyse von Algorithmen

Rekursion: Beispiel

- var a : array [0..n-1] of N_0;
- sum(i : N_0) : N_0

```
if i > n-1
  then 0
  else a[i] + sum(i+1)
end if
```

Rekursionsformen: Endrekursion

- var a : array [0..n-1] of N_0;
- sum'(i : N_0, acc: N_0) : N_0

  if i > n-1

  then acc

  else sum'(i+1, a[i] + acc)

  end if

- sum(j : N_0) : N_0

  sum'(j, 0)

- sum(0)

Akkumulator

Endrekursion ist Spezialfall des Tail-Call-Prinzips

Umwandlung von Endrekursion in eine Schleife

- $sum(j : N_0) : N_0$

  sum'(j, 0)

- sum'(i, acc: N_0) : N_0

  if i > n-1

  then acc

  else sum'(i+1, a[i] + acc)

  end if

"Rekursives Programm"

- $sum(i : N_0) : N_0$

  begin

  var acc : N_0 ;

  acc := 0 ;

  while ¬(i > n-1)

  acc := a[i] + acc ;

  i := i+1

  end while ;

  acc

  end

"Iteratives Programm"

Partielle Korrektheit von rekursiven Programmen

- Beweistechnik: Induktion
- Beispiel:
  - Linear rekursive Summenberechnung ab Position k
- Basisfall: k >= n
- Induktionsschritt: k < n

Induktion: Basisfall: k >= n

- { V } s := sum(k) { s = 0 }
- begin

```
   var i' : N0;
{ V }  i' := k;
{ V' } if i' >= n
          then { V1 } s := 0
          else { V2 } s := a[i'] + sum(i' + 1)
       end if
end
{  s =  0  }
```

Zu zeigen: V = true

Induktionsschritt k < n

- $\{ \text{sum}(k+1) = \sum_{j=k+1}^{n-1} a[j] \}$ s := sum(k) $\{ s = \sum_{j=k}^{n-1} a[j] \}$
- begin

```
   var i' : N0;
{ V }  i' := k;
{ V' } if i' >= n
          then { V1 } s := 0
          else { V2 } s := a[i'] + sum(i' + 1)
       end if
end
```

$\{ s = \sum_{j=k}^{n-1} a[j] \}$

Termination von rekursiven Programmen

- Variante t
  - Im Beispiel: $t = n - i$
- Fortschritt:
  - Es gibt ein T, so daß $\{ t > T \}$ S $\{ t = T \}$ korrekt ist
  - Im Beispiel: $\{ n - i'' > T \}$ sum(k) $\{ n - i' = T \}$
- Abbruchbedingung $\neg B$ (Wann kein rek. Aufruf?):
  - $(t = 0) \rightarrow \neg B$ gültig.
  - Im „sum"-Beispiel: $\neg B = i \geq n$ also $B = i < n$

Korrektheit des While-Programms

- $I \wedge \neg B$: $acc = \sum_{j=0}^{n-1} a[j] \wedge i = n$
- B : $i < n$
- I : $acc = \sum_{j=0}^{n-1} a[j] \wedge i \leq n$

Termination des While-Programms

- Variante t
  - Im Beispiel: $t = n - i$
- Fortschritt:
  - Es gibt ein T, so daß $\{ t > T \}$ Wh-Rumpf $\{ t = T \}$ korrekt ist
  - Im Beispiel: $\{ n - i > T \}$ acc, i := ..., i+1 $\{ n - i = T \}$
- Abbruchbedingung $\neg B$ (Wann kein rek. Aufruf?):
  - $(t = 0) \rightarrow \neg B$ gültig.
  - Im „sum"-Beispiel: $\neg B \;=\; i \geq n$ also $B = \; i < n$

Was kommt beim nächsten Mal?

- Datenstrukturen
- Algorithmen und deren Analyse