Grundlagen der Programmierung

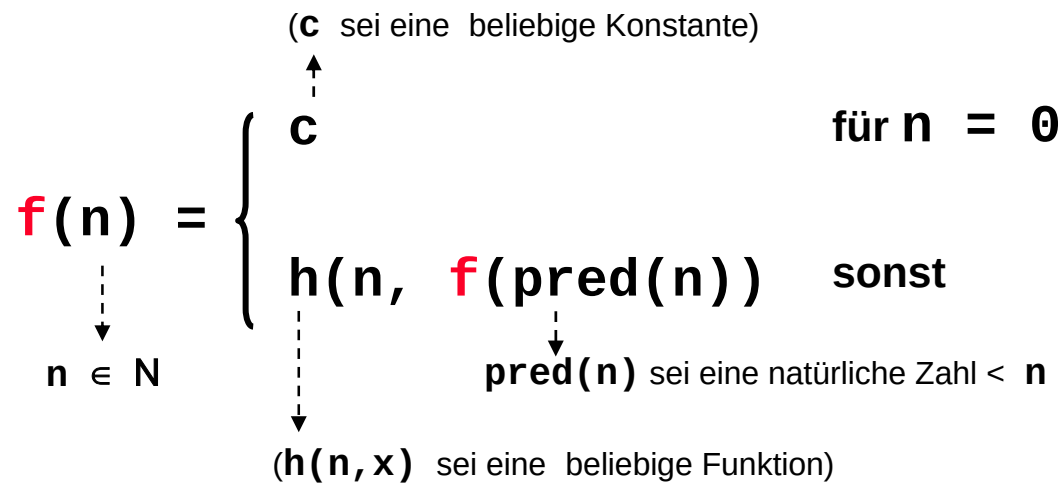
Vorlesung 10 vom 06.01.2005

Sebastian Iwanowski
FH Wedel

Grundlagen der Programmierung

1. Einführung
   - Grundlegende Eigenschaften von Algorithmen und Programmen
2. Logik
   - Aussagenlogik
   - Prädikatenlogik
3. Programmentwicklung und –verifikation
   - Grundlagen der Programmverifikation
   - Zuweisungen und Verbundanweisungen
   - Verzweigungen
   - Schleifen
   - Modularisierung
   - ➔ Rekursion
4. Entwurf und Analyse von Algorithmen
   - Klassifikation von Algorithmen
   - Programmierung von Algorithmen
   - Bewertung von Algorithmen

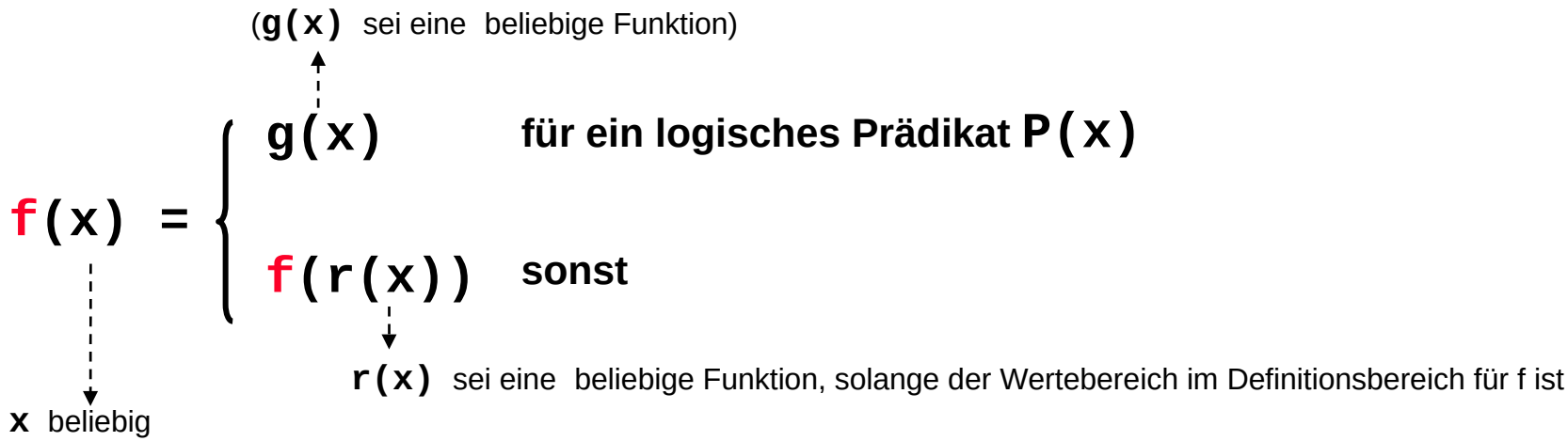
Primitiv rekursive Funktionen




```
procedure f(n: Integer): Integer
if (n=0)
  then
    return c
  else
    return h(n, f((pred(n)))
end {f}
```

Vorteil: **Terminierung ist immer gewährleistet**

Endrekursive Funktionen




```
procedure f(x): ResultType
if P(x)
  then
    return g(x)
  else
    return f(r(x))
end {f}
```

Vorteil: **Es gibt Algorithmus zur automatischen Implementierung auf dem Computer**

Linear rekursive Funktionen

$$f(x) = \begin{cases} g(x) & \text{für ein logisches Prädikat } P(x) \\ h(x, f(r(x))) & \text{sonst} \end{cases}$$

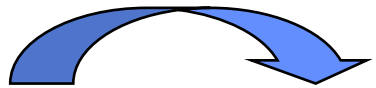
```
procedure f(x): ResultType
if P(x)
  then
    return g(x)
  else
    return h(x, f(r(x)))
end {f}
```

Primitiv rekursive Funktionen ⇘

Linear rekursive Funktionen

Endrekursive Funktionen ⇗

Transformationen



Linear rekursive Funktion → Endrekursive Funktion

geht nur unter den Voraussetzungen:

i) $\forall x,y,z: h(x, h(y,z)) = h(h(x,y), z)$

ii) $\exists e: \forall x: h(e, x) = x$

```
procedure f(x): ResultType
if P(x)
  then
    return g(x)
  else
    return h(x,f(r(x)))
end {f}
```

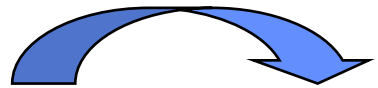
zweidimensionaler Vektor (x,a)

```
procedure fAux(x,a): ResultType
if P(x)
  then
    return h(a,g(x))
  else
    return fAux(r(x),h(a,x)))
end {fAux}
```

„rAux“: Funktion von (x,a)

```
procedure f(x): ResultType
  return fAux(x,e)
end {f}
```

Transformationen



Endrekursive Funktion **Schleife**

geht immer !

```
procedure f(x): ResultType
if P(x)
  then
    return g(x)
  else
    return f(r(x))
end {f}
```

```
procedure f(x): ResultType
while ¬P(x)do
  x := r(x);
return g(x)
end {f}
```

Allgemeine rekursive Funktionen

Fibonacci-Funktion:

$$f(n) = \begin{cases} 1 & \text{für } n \leq 1 \\ f(n-2) + f(n-1) & \text{sonst} \end{cases}$$

McCarthy-Funktion:

$$f(n) = \begin{cases} n-10 & \text{für } n > 100 \\ f(f(n+11)) & \text{sonst} \end{cases}$$

Ulam-Collatz-Funktion:

$$f(n) = \begin{cases} 1 & \text{für } n = 1 \\ f(n)/2 & \text{für gerade } n \\ f(3n+1) & \text{für ungerade } n \end{cases}$$

Allgemeine rekursive Funktionen

- **Nicht jede rekursive Funktion ist linear rekursiv oder endrekursiv**
- **Alle rekursiven Funktionen können auch nichtrekursiv formuliert werden.**

Was ist besser: Rekursive oder iterative Formulierung ?

Was ist für die Klausur relevant ?

Vorlesung 1-2: Überblick / Grundlegende Eigenschaften

Grundlegende Begriffe und Konzepte: Funktion, Spezifikation, Algorithmus, Programm, Verifikation, Konstruktion und alles, was damit zusammenhängt. Fähigkeit, die Konzepte an Minibeispielen anzuwenden.

Vorlesung 2-4: Aussagenlogik

Grundlegende Begriffe wie Aussage, Wahrheitswert, Äquivalenzregel, Schlussregel, Formel, Belegung, Erfüllbarkeit. Sicherer Umgang mit aussagenlogischen Umformungen, eindeutige und exakte Notation. Boolesche Algebren: Zusammenhang zwischen Aussagenlogik und Mengenlehre. Konjunktive Normalform: Begriffe und Fähigkeit der Umformung in eine solche.

Vorlesung 4-5: Prädikatenlogik

Unterschied zur Aussagenlogik: Variable, Prädikate, Funktionen, Quantoren. Erklären dieser Begriffe an Beispielen, aktive und passive Beherrschung der Notation. Rechenregeln für Quantoren, Anwendung der Prädikatenlogik auf Minibeispiele aus dem täglichen Leben und der Arithmetik.

Was ist für die Klausur relevant ?

Vorlesung 6-7: Grundlagen der Programmverifikation

Hoare-Tripel: Aktive und passive Beherrschung der Notation. Zusammenhang zwischen unterschiedlichen Stärken von Vorbedingungen und Nachbedingungen, Exakte Verifikation kleinster Programmteile: Finden von stärksten Nachbedingungen und schwächsten Vorbedingungen.

Vorlesung 7: Verzweigungen

Finden von stärksten Nachbedingungen und schwächsten Vorbedingungen in Anwendungsbeispielen.

Vorlesung 8-9: Schleifen

Bestimmung und Verifikation von Invarianten sowie von Varianten für die Terminierung, Verifikation von kleinen Schleifen entweder mit vollständiger Induktion oder mit dem Lösungsverfahren 1) bis 3) zur Konstruktion.

Was ist für die Klausur relevant ?

Vorlesung 9: Modularisierung

Parameter, Rückgabewerte: Erklärung der Bedeutung und Verwendung, Anwendung in Minibeispielen. Call-by-reference und call-by-value: Erklärung der Unterschiede, Vorteile und Nachteile der beiden Parameterübergabetechniken.

Vorlesung 9: Rekursion

Nachvollziehen des Programmablaufs vorgegebener rekursiver Programme, Bestimmen von notwendigen Vorbedingungen.

Nur für die „normale“ Klausur in den Studiengängen MInf und WInf:

Vorlesung 10: Rekursion

Erklärung der Konzepte primitiv rekursiv, endrekursiv und linear rekursiv. Erkennen dieser Konzepte an Beispielen. Umformung von endrekursiven Programmen in Schleifen. Vor- und Nachteile der rekursiven Formulierung gegenüber der iterativen.

Vorlesung 11-12: Entwurf und Analyse von Algorithmen

wird am Ende von Vorlesung 12 bekannt gegeben

Beim nächsten Mal:

12.01.: ***Übergangsklausur für Inf und TInf***

Viel Erfolg !

13.01.: ***Entwurf und Analyse von Algorithmen***