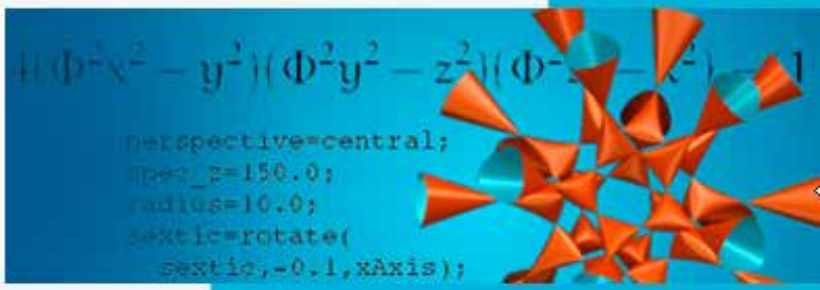
Philipps Universität Marburg

Deklarative Programmierung

Sommersemester 2018

Prof. Christoph Bockisch
(Programmiersprachen und –werkzeuge)
Steffen Dick, Alexander Bille, Johannes Frankenau,
Patrick Frömel, Niclas Schmidt, Jonas Stettin,
Robert Tran, Julian Velten



Regeln

- Aufgaben
  - Jede Aufgabe wird für 6 Minuten lang eingeblendet
  - Ein Pfiff macht Sie auf den Wechsel aufmerksam
- Lösungen
  - Teamarbeit: max. 3er Teams
  - Jede Gruppe erhält ein Formular zum Eintragen der Lösungen
  - Tragen Sie auf das Formular alle Teammitglieder ein
  - Kreuzen Sie für jede Aufgabe die jeweils korrekten Antworten an (**es können auch mehrere Lösungen korrekt sein!**)
- Bewertung
  - Im Anschluss geben Sie die Formulare ab
  - Die Tutoren korrigieren diese während wir die Lösungen besprechen
  - Es gibt 8 Aufgaben
  - Eine Lösung gilt nur als korrekt, wenn **alle korrekten Lösungen** angekreuzt sind und **keine falsche Lösung** angekreuzt ist
- Preis
  - Pro korrekter Antwort ein Punkt

Keine Hilfsmittel erlaubt, außer Stift und Papier.

Aufgabe 1

- Was ist die minimale Anzahl an Reduktionen, um den folgenden Ausdruck zu einem Wert zu reduzieren?

```
(f (g 2))
```

a) 8
b) 7
c) 5
d) 10

Umgebung

```
(define f
  (lambda (x)
    (cond
      [(> 0 x) (/ 2 x)]
      [true (* 2 x)])))
(define g
    (lambda (x) x))
```

Aufgabe 2

- Welche der Möglichkeiten sind gültige Reduktionen des folgenden Ausdrucks?

```
(posn-y
  (make-posn `(+ a b) (+ a b)))
```

Umgebung
```
(define a 42)
(define b 21)
```

a) `(posn-y (make-posn (list a b) 63))`

b) **63**

c) `(posn-y (make-posn (list 63) (+ a b)))`

d) Keine der Antworten, denn es gibt einen Typfeher

Aufgabe 3

```
; Ein Strecke ist eins von
; - “kurz”
; - ein Number zwischen 1 und 5 (einschließlich)
; interp. Die Kategorie für ein Nahverkehrsticket:
; entweder eine Kurzstrecke, oder eine Strecke innerhalb Tarifzone 1, 2, 3, 4 oder 5

; Ein Zeit ist ein String
; interp. Datum und Uhrzeit nach deutschem Gebietsschema codiert
(define SUMMER_START "25.03.2018 02:00")

(define-struct ticket (zeit strecke))
; Ein Ticket ist ein (make-ticket String Strecke).
; interp. Ein Ticket stellt die Zeit, zu der
; es entwertet wurde und eine Strecke
; auf der es gültig ist dar

; Ticket -> String
; Gibt eine textuelle Darstellung
; der Informationen des Tickets
; zurück
(define (print ticket) …)
```

- Wieviele Testfällt werden für die Funktion print mindestens benötigt?

a) 5
b) 6
c) 12
d) 13

Aufgabe 4

- Welche der folgenden Ausdrücke haben eine Bedeutung in der ISL+ Sprache?

a) `(second (third '(1 (2 3) 4 (5 6))))`

b) `(length `(a b c (make-posn d e) f g))`

c) `(beside (circle 10 "solid" "red") )`

d) `((lambda (x) (cond [true 1] [(< x 6) 2])) 3)`

Aufgabe 5

- Induktionsbeweis

1. **Behauptung**: Für jede natürliche Zahl n gilt die Aussage B(n): Ist unter n Tieren eines ein Elefant, so sind alle n Tiere Elefanten.
2. **Basisfall**: Die Aussage B(1) ist trivial
3. **Wir dürfen verwenden**: Sei n eine natürliche Zahl und sei B(n) wahr

- **Induktionsschritt**:

4. Wir stellen die n+1 Tiere so in einer Reihe auf, dass der Elefant an erster Stelle steht. → Unter den ersten n Tieren gibt es einen Elefanten. Induktionsannahme → Die ersten n Tiere in der Reihe sind Elefanten.
5. Damit befindet aber auch unter den letzten n Tieren ein Elefant. Induktionsannahme → die letzten n Tiere sind Elefanten. → Alle n + 1 Tiere sind Elefanten.

a) Der Beweis ist korrekt
b) Es befindet sich ein Fehler in Schritt 1
c) Es befindet sich ein Fehler in Schritt 2
d) Es befindet sich ein Fehler in Schritt 3
e) Es befindet sich ein Fehler in Schritt 4
f) Es befindet sich ein Fehler in Schritt 5

Aufgabe 6

```
(define (a b c d)
  (cond
    [(empty? b) c]
    [(empty? c) empty]
    [else (cons
      (d (first b) (first c))
      (a (rest b) (rest c) d))]
  )
)
```

- Wie viele Typparameter werden für die Signatur der Funktion h benötigt?

a) 4

b) 1

c) 3

d) 2

Beispielsweise liefert die Funktion für den Aufruf:
(a '(1 2 3) '(5 6 7) (lambda (x y) (+ y x))
das Ergebnis: (list 6 8 10)

Aufgabe 7

```
(define (flatten lst)
  (cond
    [(empty? lst) empty]
    [(list? (first lst))
      (append (flatten (first lst)) (flatten (rest lst)))]
    [else
      (cons (first lst) (flatten (rest lst)))]))
```

- Wie häufig wird die Funktion flatten aufgarufen, wenn sie mit folgendem Ausdruck aufgerufen wird? (zählen Sie den initialen und alle rekursiven Aufrufe)
  `(flatten '(a (b c) (d (e)) f))`

a) 11 mal
b) 13 mal
c) 19 mal
d) unendlich oft

Aufgabe 8

- Betrachten Sie folgende Funktion, die einen Baum in Form einer S-Expression als Argument erwartet

```
(define (f tree)
  (if (empty? tree)
    0
    (if (list? tree)
      (+ (f (first tree)) (f (rest tree)))
      1
    )
  )
)
```

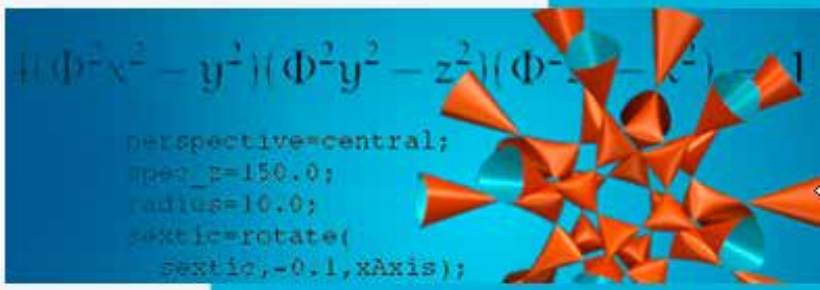
Was berechnet diese Funktion?

a) Wie viele Kind-Knoten ein innerer Knoten höchstens hat.
b) Wie viele Eltern-Knoten ein innerer Knoten höchstens hat.
c) Die Anzahl von Blatt-Knoten in dem Teilbaum.
d) Die Anzahl der Kanten in dem Baum.

Philipps Universität Marburg

ENDE

Nachfolgend: Besprechung



Aufgabe 1

- Was ist die minimale Anzahl an Reduktionen, um den folgenden Ausdruck zu einem Wert zu reduzieren?

```
(f (g 2))
```

a) 8
b) 7
c) 5
d) 10

Umgebung

```
(define f
  (lambda (x)
    (cond
      [(> 0 x) (/ 2 x)]
      [true (* 2 x)])))
(define g
    (lambda (x) x))
```

Aufgabe 1

Umgebung

```
(define f
  (lambda (x)
    (cond
      [(> 0 x) (/ 2 x)]
      [true (* 2 x)])))
(define g
    (lambda (x) x))
```

```
(f (g 2))
→ CONST
(f (((lambda (x) x) 2))
→ APP
(f 2)
→ CONST
((lambda (x) (cond [(> 0 x) (/ 2 x)] [true (* 2 x)])) 2)
→ APP
(cond [(> 0 2) (/ 2 2)] [true (* 2 2)])
→ PRIM
(cond [false (/ 2 2)] [true (* 2 2)])
→ CONT-false
(cond [true (* 2 2)])
→ COND-true
(* 2 2)
→ PRIM
4
```

a) 8 Reduktionen

Aufgabe 2

- Welche der Möglichkeiten sind gültige Reduktionen des folgenden Ausdrucks?

```
(posn-y
  (make-posn `(+ a b) (+ a b)))
```

Umgebung
```
(define a 42)
(define b 21)
```

a) `(posn-y (make-posn (list a b) 63))`

b) **63**

c) `(posn-y (make-posn (list 63) (+ a b)))`

d) Keine der Antworten, denn es gibt einen Typfeher

Aufgabe 2

(**posn-y** (**make-posn** `(+ a b) (+ a b)))

Desugar:

(**posn-y** (**make-posn** (list `+ `a `b) (+ a b)))

→ CONST (2x)

(**posn-y** (**make-posn** (list `+ `a `b) (+ 42 21)))

→ PRIM

(**posn-y** (**make-posn** (list `+ `a `b) 63))

→ STRUCT-make

(posn-y <make-posn (list `+ `a `b) 63>)

→ STRUCT-select

63

Umgebung
(**define** **a** 42)
(**define** **b** 21)

a) ~~(posn-y (make-posn (list a b) 63))~~
b) 63
c) ~~(posn-y (make-posn (list 63) (+ a b)))~~
d) ~~Keine der Antworten, denn es gibt einen Typfehler~~

Aufgabe 3

```
; Ein Strecke ist eins von
; - “kurz”
; - ein Number zwischen 1 und 5 (einschließlich)
; interp. Die Kategorie für ein Nahverkehrsticket:
; entweder eine Kurzstrecke, oder eine Strecke innerhalb Tarifzone 1, 2, 3, 4 oder 5

; Ein Zeit ist ein String
; interp. Datum und Uhrzeit nach deutschem Gebietsschema codiert
(define SUMMER_START "25.03.2018 02:00")

(define-struct ticket (zeit strecke))
; Ein Ticket ist ein (make-ticket String Strecke).
; interp. Ein Ticket stellt die Zeit, zu der
; es entwertet wurde und eine Strecke
; auf der es gültig ist dar

; Ticket -> String
; Gibt eine textuelle Darstellung
; der Informationen des Tickets
; zurück
(define (print ticket) …)
```

- Wieviele Testfällt werden für die Funktion print mindestens benötigt?

a) 5
b) 6
c) 12
d) 13

Aufgabe 3

```
; Ein Strecke ist eins von
; - “kurz”
; - ein Number zwischen 1 und 5 (einschließlich)
; interp. Die Kategorie für ein Nahverkehrsticket:
; entweder eine Kurzstrecke, oder eine Strecke innerhalb Ta
```

Strecke: Summentyp bestehend aus 1 Stringwert und einem Enumerationstyp mit 5 Varianten

```
; Ein Zeit ist ein String
; interp. Datum und Uhrzeit nach deutschem Gebietsschem
(define SUMMER_START "25.03.2018 02:00")
```

Zeit: nur ein Beispielwert

```
(define-struct ticket (zeit strecke))
; Ein Ticket ist ein (make-ticket String Strecke).
; interp. Ein Ticket stellt die Zeit, zu der
; es entwertet wurde und eine Strecke
; auf der es gültig ist dar

; Ticket -> String
; Gibt eine textuelle Darstellung
; der Informationen des Tickets
; zurück
(define (print ticket) …)
```

- Wiev
  Funktion print mindestens benötigt?

a) 5
b) 6
c) 1
d) 1

Ticket: alle Kombinationen der sinnvollen Repräsentanten von Strecke (6) und Zeit (1): Mind. 6 Kombinationen.

Antwort b)

Aufgabe 4

- Welche der folgenden Ausdrücke haben eine Bedeutung in der ISL+ Sprache?

a) `(second (third '(1 (2 3) 4 (5 6))))`

b) `(length `(a b c (make-posn d e) f g))`

c) `(beside (circle 10 "solid" "red") )`

d) `((lambda (x) (cond [true 1] [(< x 6) 2])) 3)`

Aufgabe 4

- Welche der folgenden Ausdrücke haben eine Bedeutung in der ISL+ Sprache?

a) (second (third '(1 (2 3) 4 (5 6))))

Fehler: second ist auf Listen definiert, es wird aber auf 4 angewandt.

b) (length `(a b c (make-posn d e) f g))

6

c) (beside (circle 10 "solid" "red"))

d) ((lambda (x) (cond [true 1] [(< x 6) 2])) 3)

Stärke Bedingung vor schwächerer. Ergebnis vielleicht unerwünscht, aber gültiges Programm (Ergebnis 1).

korrekt: b, c, d

Aufgabe 5

- Induktionsbeweis

1. **Behauptung**: Für jede natürliche Zahl n gilt die Aussage B(n): Ist unter n Tieren eines ein Elefant, so sind alle n Tiere Elefanten.
2. **Basisfall**: Die Aussage B(1) ist trivial
3. **Wir dürfen verwenden**: Sei n eine natürliche Zahl und sei B(n) wahr

- **Induktionsschritt**:

4. Wir stellen die n+1 Tiere so in einer Reihe auf, dass der Elefant an erster Stelle steht. → Unter den ersten n Tieren gibt es einen Elefanten. Induktionsannahme → Die ersten n Tiere in der Reihe sind Elefanten.
5. Damit befindet aber auch unter den letzten n Tieren ein Elefant. Induktionsannahme → die letzten n Tiere sind Elefanten. → Alle n + 1 Tiere sind Elefanten.

a) Der Beweis ist korrekt
b) Es befindet sich ein Fehler in Schritt 1
c) Es befindet sich ein Fehler in Schritt 2
d) Es befindet sich ein Fehler in Schritt 3
e) Es befindet sich ein Fehler in Schritt 4
f) Es befindet sich ein Fehler in Schritt 5

Aufgab

Die Behauptung ist natürlich falsch, daher kann auch der Schritt 1 als Fehlerhaft angesehen werden. Allerdings folgt die Definition der Behauptung dem Muster für Induktionsbeweise und stellt noch keine falsche Schlussfolgerung dar.

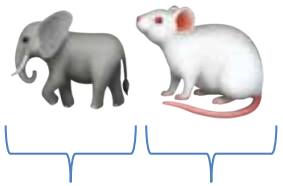
- Induktionsbew

1. **Behauptung**: Für jede natürliche Zahl n gilt die Aussage B(n): Ist unter n Tieren eines ein Elefant, so sind alle n Tiere Elefanten.
2. **Basisfall**: Die Aussage B(1) ist trivial
3. **Wir dürfen verwenden**: Sei n eine natürliche Zahl und sei B(n) wahr

- **Induktionsschritt**:

4. Wir stellen die n+1 Tiere so in einer Reihe auf, dass der Elefant an erster Stelle steht. → Unter den ersten n Tieren gibt es einen Elefanten. Induktionsannahme → Die ersten n Tiere in der Reihe sind Elefanten.
5. Damit befindet aber auch unter den letzten n Tieren ein Elefant. Induktionsannahme → die letzten n Tiere sind Elefanten. → Alle n + 1 Tiere sind Elefanten.

n+1 = 2:



n n

mind. 1 Elefant enthalten
→ mit B(n): alle sind Elefanten

Annahme aus Schritt 5
bei n + 1 = 2 nicht erfüllt

f) Es befindet sich eine Fehler in Schritt 5

Aufgabe 6

```
(define (a b c d)
  (cond
    [(empty? b) c]
    [(empty? c) empty]
    [else (cons
      (d (first b) (first c))
      (a (rest b) (rest c) d))]
  )
)
```

- Wie viele Typparameter werden für die Signatur der Funktion h benötigt?

a) 4

b) 1

c) 3

d) 2

Beispielsweise liefert die Funktion für den Aufruf:
(a '(1 2 3) '(5 6 7) (lambda (x y) (+ y x))
das Ergebnis: (list 6 8 10)

d muss eine Closure sein, der Rückgabewert kann als Element im Ergebnis dienen. Elemente vo von b und c sind Argumente
(list-of X) (list-of Y) (X Y -> Y) -> (list-of Y)

```
[(      b) c]
[(e    y? c) empty]
[els  (cons
  (d (first b) (first c))
  (a (rest b) (rest c) d))]
)
)
```

b und c müssen Listen sein
(list-of X) (list-of Y) *Z* -> *U*

B

(a (1 2 3) (5 6 7) (lambd

das Ergebnis: (list 6 8 10)

Wie viele Typparameter werden für die Signatur der Funktion h benötigt?

c kann der Rückgabewert sein
(list-of X) (list-of Y) *Z* -> (list-of Y)

c) 3

d) 2

[X,Y] (list-of X) (list-of Y) (X Y -> Y) -> (list-of Y)
Antwort: d) 2

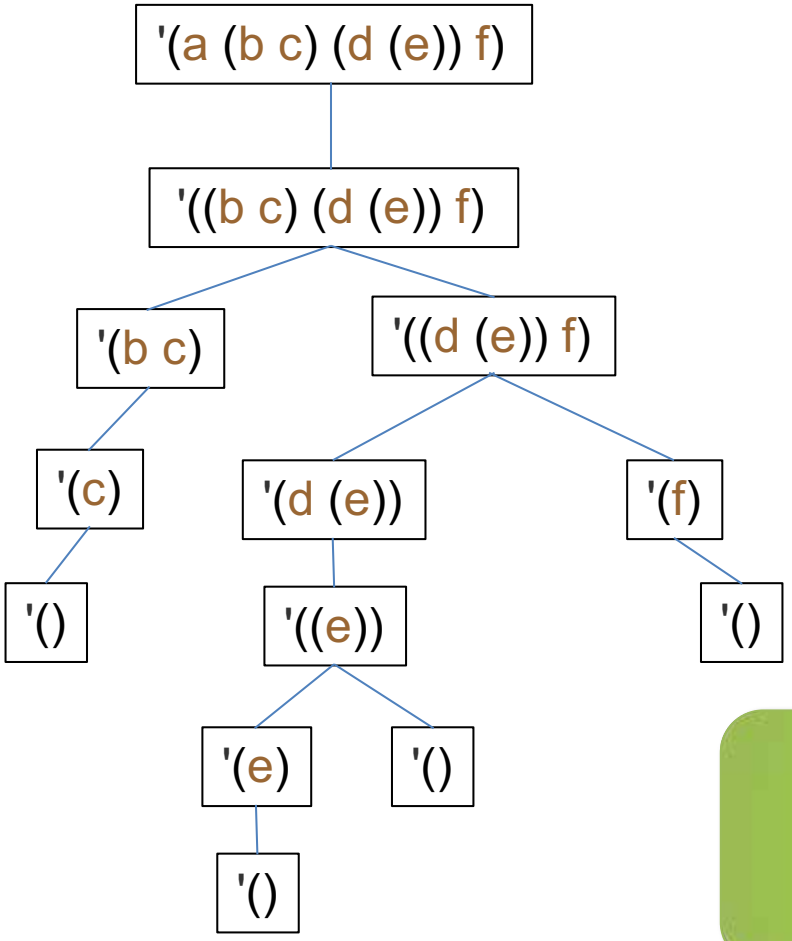
Aufgabe 7

```
(define (flatten lst)
  (cond
    [(empty? lst) empty]
    [(list? (first lst))
      (append (flatten (first lst)) (flatten (rest lst)))]
    [else
      (cons (first lst) (flatten (rest lst)))]))
```

- Wie häufig wird die Funktion flatten aufgerufen, wenn sie mit folgendem Ausdruck aufgerufen wird? (zählen Sie den initialen und alle rekursiven Aufrufe)
  `(flatten '(a (b c) (d (e)) f))`

a) 11 mal
b) 13 mal
c) 19 mal
d) unendlich oft

Aufgabe 7



Antwort: b) 13 Mal

Aufgabe 8

- Betrachten Sie folgende Funktion, die einen Baum in Form einer S-Expression als Argument erwartet

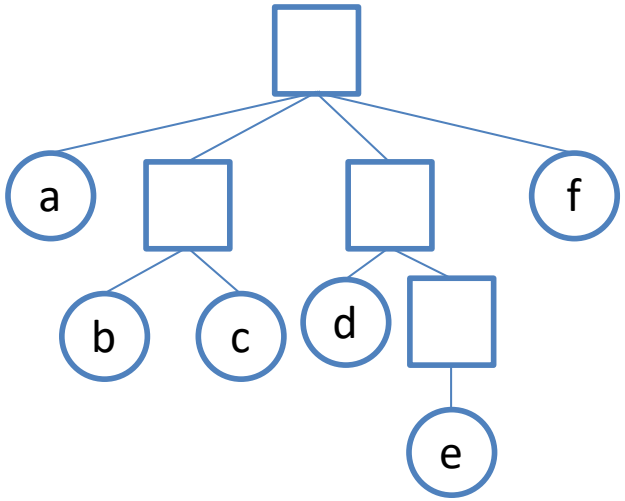
```
(define (f tree)
  (if (empty? tree)
    0
    (if (list? tree)
      (+ (f (first tree)) (f (rest tree)))
      1
    )
  )
)
```

Was berechnet diese Funktion?

a) Wie viele Kind-Knoten ein innerer Knoten höchstens hat.
b) Wie viele Eltern-Knoten ein innerer Knoten höchstens hat.
c) Die Anzahl von Blatt-Knoten in dem Teilbaum.
d) Die Anzahl der Kanten in dem Baum.

Aufgabe 8

- Baum als S-Expression
- Beispiel
- '(a (b c) (d (e)) f)




```
(define (f tree)
  (if (empty? tree)
    0
    (if (list? tree)
      (+ (f (first tree)) (f (rest tree)))
      1
    )
  )
)
```

Antwort c:
Die Anzahl von Blatt-Knoten in dem Teilbaum.