



INFORMATIK I

Tutorium 08.11.2016

BESPRECHUNG

Blatt 2



WIEDERHOLUNG

.....
Vorlesung & Für Blatt 3



SUBSTITUTIONSMODELL

- Literale reduzieren zu sich selbst (Bsp.: 1, #t, „abc“) —> eval_lit
- Identifier reduzieren zu dem Wert an den sie gebunden sind (Bsp.: pi, kilometers-per-mile, etc) —> eval_id
- Lambda-Abstraktion keine Reduktion möglich —> eval_λ
- Eine Applikation (f e1 e2) reduziert zu (f' e1' e2') (jeweils reduzieren). Falls f' prim. Operation, wende f' auf e1' e2' an —> apply_prim, ansonsten Argumentwerte e1' e2' in Rumpf von f' einsetzen —> apply_lambda

SUBSTITUTIONSMODELL – BEISPIELE

- $(+ \ 40 \ 2) \longrightarrow \text{eval_id} \ (\#<\text{procedure:}+> \ 40 \ 2) \longrightarrow 2x \ \text{eval_lit} \ (\#<\text{procedure:}+> \ 40 \ 2) \longrightarrow \text{apply_prim} \ (42)$
- $(\text{sqr} \ 9) \longrightarrow \text{eval_id} \ ((\text{lambda} \ (x) \ (* \ x \ x)) \ 9) \longrightarrow \text{eval_lit} \ ((\text{lambda} \ (x) \ (* \ x \ x)) \ 9) \longrightarrow \text{apply_}\lambda \ (* \ 9 \ 9) \longrightarrow \text{eval_id} \ (\#<\text{procedure:}*> \ 9 \ 9)) \longrightarrow 2x \ \text{eval_lit} \longrightarrow \text{apply_prim} \ (81)$
- Am besten über den Stepper in Dr.Racket machen!

BINDUNG VON VARIABLEN

- Lexikalische Bindung von innen nach außen
- Zu beachten bei gleichnamigen Variablen in verschachtelten Funktionen!

```
((lambda (r)
  (* pi
    ((lambda (r) (* r r)) r)))
10)
```

- Zwei verschiedene r —> siehe Stepper

FALLUNTERSCHIEDUNGEN – COND

- `(cond (<t1> <e1>) ... (<t_n> <e_n>) (else e_n+1))`
- Beispiel:

```
(define compare
  (lambda (a)
    (cond ((< a 0) "a ist kleiner 0")
          ((> a 0) "a ist größer 0")
          (else "a ist gleich 0"))))
```

FALLUNTERSCHIEDUNG – IF

➤ (if <t_1> <e_1> <e_2>)

➤ Beispiel:

```
(define compare1
  (lambda (a)
    (if (<= a 0)
        "a ist kleiner gleich 0"
        "a ist größer 0")))
```


OR UND AND

- $(\text{or } \langle t_1 \rangle \langle t_2 \rangle \dots \langle t_n \rangle)$ evaluiert zu $\#t$, wenn eine der Bedingungen erfüllt ist, ansonsten zu $\#f$
- $(\text{and } \langle t_1 \rangle \langle t_2 \rangle \dots \langle t_n \rangle)$ evaluiert zu $\#t$, wenn alle Bedingungen erfüllt sind, ansonsten zu $\#f$

ÜBUNGSAUFGABEN

- Schreibe eine Prozedur `imply` die die Implikation (logische Funktion) implementiert. Sie erhält zwei Parameter `a`, `b` (jeweils `#f` oder `#t`) und bestimmt $a \rightarrow b$
- Schreibe eine Prozedur `xor`, die das XOR (logische Funktion) implementiert. Sie erhält zwei Parameter `a`, `b` (jeweils `#f` oder `#t`) und bestimmt $a \text{ XOR } b$
- Schreibe eine Prozedur `pow`, die $f(x,y) = x^y$ berechnet
- Schreibe eine Prozedur `sum`, die Summe von 1 bis `n` berechnet (mathematisch und rekursiv)