

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

# Informatik I Vorlesung

Wintersemester 2016/2017

Mitschrieb von  
Julian Wolff

Aktueller Stand 28. Oktober 2016

# Inhaltsverzeichnis

<b>1</b>	<b>Scheme: Ausdrücke, Auswertung und Abstraktion</b>	<b>2</b>
1.1	REPL . . . . .	2
1.2	Literale . . . . .	2
1.3	Zusammengesetzte Ausdrücke . . . . .	3
1.4	Identifizier . . . . .	3
1.5	Lambda-Abstraktion . . . . .	3
1.6	Kommentare . . . . .	4
1.7	Signaturen . . . . .	4
1.8	Prozedur-Signaturen . . . . .	5
1.9	Testfälle . . . . .	5

# Scheme: Ausdrücke, Auswertung und Abstraktion

## REPL

Definition	DrRacket
Interaktion	REPL

Die Anwendung von Funktionen wird in Scheme ausschließlich in Präfixnotation durchgeführt:

Mathematik	Scheme
44-2	(-44 2)
f(x,y)	(f x y)
$\sqrt{81}$	(sqrt 81)
$\lfloor x \rfloor$	(floor x)
$9^2$	(expt 9 2)
3!	(! 3)

Allgemein:  $\langle \text{Funktion} \rangle \langle \text{argument} \rangle$

(+ 402) und (odd? 42) sind Beispiele für die Ausdrücke, die bei Auswertung einen Wert liefern. (Notation  $\rightsquigarrow$ ) heißt Auswertung/Evaluation/Reduktion.

(+ 40 2)  $\rightsquigarrow$  42  
 (add? 42)  $\rightsquigarrow$  #f  
Eval Eval

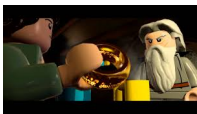
Interaktionsfenster:

Read  $\rightsquigarrow$  Eval  $\rightsquigarrow$  Print  
Loop

REPL

## Literale

Literale stehen für einen konstanten Wert (auch: Konstante) und sind nicht weiter reduzierbar.

<u>Literal</u>		<u>Signatur</u>
#t #f	(true, false, Wahrheitswerte)	boolean
„ac“ „x“ „“	(Zeichenketten)	string
0 1904 -42 007	(ganze Zahlen)	integer
0.42 3.1415 -273.15	(Fließkommazahlen)	real
1/2 3/4 -1/10	(rationale Zahlen)	rational
	(Bilder)	image


## Zusammengesetzte Ausdrücke

Auswertung zusammengesetzte Ausdrücke (composite expression) in mehreren Schritten (Steps), "von innen nach außen", bis keine weitere Reduktion möglich ist:

$$(+ (+20\ 20) (+\ 1\ 1)) \rightsquigarrow (+\ 40\ (+\ 1\ 1)) \rightsquigarrow (+\ 40\ 2) \rightsquigarrow 42$$


Beispiel:

$$0.7 + (\tfrac{1}{2}/0.25) - (0.6/0.3) = 0.7$$

 Achtung: Scheme rundet bei Arithmetik mit Fließkommazahlen (interne Darstellung nicht präzise). Die Arithmetik mit rationalen Zahlen ist exakt.

## Identifizier

Ein Wert kann an einen Namen (identifier) gebunden werden, durch `(define<id><expression>)`. Es erlaubt konsistente Wiederverwendung und dient der Selbstdokumentation von Programmen.

 Achtung: Dies ist eine Spezialform und kein Ausdruck. Insbesondere besitzt diese Spezialform keinen Wert, sondern einen Effekt: der Name `<id>` wird durch den Wert von `<expression>` gebunden. Namen können in Scheme fast beliebig gewählt werden, solange

- die Zeichen `()[]{}",';#\` nicht vorkommen
- der name nicht einem numerischen Literal gleicht
- keinen Whitespaße (Leerzeichen, Tabulatoren, Neuwlines) enthalten sind

Beispiel: `Euro`  $\rightarrow$  `US-$`

 Achtung: Groß-/Kleinschreibung ist in Identifiern nicht relevant.

## Lambda-Abstraktion

Eine Lambda-Abstraktion (auch: Funktion, Prozedur) erlaubt die Formulierung von Ausdrücken, in denen mittels Parametern von konkreten Werten abstrahiert wird:

$$(\text{lambda } (\langle p_1 \rangle \langle p_2 \rangle \dots) \langle \text{expr} \rangle)$$

`expr` ist der Rumpf und enthält Vorkommen der Parameter `<pi>`.

(lambda...) ist eine Spezialform. Der Wert der Lambda-Abstraktion #⟨procedure⟩  
Die Anwendung (auch: Applikation) der Lambda-Abstraktion führt zur Ersetzung  
aller Vorkommen der Parameter im Rumpf durch die angegebenen konkreten Argumente:

```
(lambda (days)(*days(*155 minutes-in-a-day)) 365)
↪ (über pfeil ! Zeichen) (*365 ( 155 minutes-in-a-day))
↪ ...
↪ 81468000
```

## Kommentare

In Scheme leitet ein Semikolon einen Kommentar ein, der bis zum Zeilenende reicht  
und von Racket bei der Auswertung ignoriert wird.

Prozeduren/Funktionen sollen im Programm eine ein- bis zweizeilige Kurzbeschreibung  
vorangestellt werden.


## Signaturen

Eine Signatur prüft, ob ein Name ⟨id⟩ an einen Wert einer angegebenen Sorte ge-  
bunden wird. Signaturverletzungen werden protokolliert.

(: ⟨ id ⟩ ⟨signatur⟩)

Bereits eingebundene Signaturen sind:

- natural     $\mathbb{N}$
- ingeger     $\mathbb{Z}$
- rational    $\mathbb{Q}$
- real        $\mathbb{R}$
- number     $\mathbb{C}$
- boolean
- string
- image

 Der Doppelpunkt „:“ ist eine Spezialform und hat daher keinen Wert, aber  
einen Effekt: Eine Signaturprüfung wird durchgeführt.

## Prozedur-Signaturen

Prozedur-Signaturen spezifizieren Signaturen sowohl für die Parameter  $\langle p_1 \rangle, \langle p_2 \rangle, \dots$  als auch für den Ergebniswert der Prozedur:

$(:\langle \text{id} \rangle(\langle \text{signatur-}p_1 \rangle \langle \text{signatur-}p_2 \rangle \dots \rightarrow \langle \text{signatur-ergebnis} \rangle))$

Prozedur-Signaturen werden bei jeder Anwendung der Funktion  $\langle \text{id} \rangle$  auf Verletzung geprüft.

## Testfälle

Testfälle dokumentieren das erwartende Ergebnis einer Prozedur für ausgewählte Argumente:

$(\text{check-expect } \langle e_1 \rangle \langle e_2 \rangle)$

Werte den Ausdruck  $\langle e_1 \rangle$  aus und teste, ob der erhaltene Wert der Erwartung (=Wert des Ausdruck  $\langle e_2 \rangle$ ) entspricht.

Einer Prozedurdefinition sollten Testfälle direkt vorangestellt werden.

$\triangle!$  „check-expect“ ist eine Spezialform und hat daher keinen Wert. Eine Testverletzung wird als Effekt protokolliert.