

# Informatik II Skript

Steffen Lindner

April 16, 2015

# Contents

<b>1</b>	<b>Einführung - 14.04.15</b>	<b>3</b>
<b>2</b>	<b>Ausdrücke, Defines, usw. - 16.04.2015</b>	<b>4</b>

# Chapter 1

## Einführung - 14.04.15

Scheme: Ausdrücke, Auswertung und Abstraktion

Dr.Racket: Definitionsfenster (oberer Bereich), Interaktionsfenster (unterer Bereich)

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

### Beispiele

Mathematik	Scheme
$44-2$	<code>(- 44 2)</code>
$f(x,y)$	<code>(f x y)</code>
$\sqrt{81}$	<code>(sqrt 81)</code>
$9^2$	<code>(expt 9 2)</code>
$3!$	<code>(! 3)</code>

Allgemein: `(< function > < arg1 > < arg2 > ...)`

`(+ 40 2)` und `(odd? 42)` sind Beispiele für Ausdrücke, die bei Auswertung einen Wert liefern. (Notation:  $\rightsquigarrow$ )

`(+ 40 2)  $\rightsquigarrow$  42` ( $\rightsquigarrow$  = Auswertng / Reduktion / Evaluation)

`(odd? 42)  $\rightsquigarrow$  #f`

Interaktionsfenster: `Read  $\rightarrow$  Eval  $\rightarrow$  Print  $\rightarrow$  Read ...` (Read-Eval-Print-Loop aka. REPL)

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

Literal:

`#t`, `#f` (true, false, Wahrheitswerte) (boolean)

`"abc"`, `"x"`, `" "` (Zeichenkette) (String)

`0` `1904` `42` `-2` (ganze Zahlen) (Integer)

`0.42` `3.1415` (Fließkommazahl) (Reel)

`1/2`, `3/4` (rationale Zahl) (Rational)

`\_("\_")\_/"` (Bilder) (Image)

## Chapter 2

# Ausdrücke, Defines, usw. - 16.04.2015

Auswertung zusammengesetzter Ausdrücke in mehreren Schritten (steps), von "innen nach außen" bis keine Reduktion mehr möglich ist.

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

Achtung: Scheme rundet bei Arithmetik mit Fließkommazahlen (interne Darstellung ist binär).

Bsp.: Auswertung des zusammengesetzten Ausdrucks  $0.7 + (1/2)/0.25 - 0.6/0.3$

Arithmetik mit rationalen Zahlen ist exakt.

Ein Wert kann an einen Namen (auch Identifier) gebunden werden, durch

$$(\text{define } \langle id \rangle \langle e \rangle) \quad (\langle id \rangle \text{ Identifier, } \langle e \rangle \text{ Expression})$$

Erlaubt konsistente Wiederverwendung und dient der Selbstdokumentation von Programmen.

Achtung: Dies ist eine sogenannte Spezifikation und kein Ausdruck. Insbesondere besitzt diese Spezialform keinen Wert, sondern einen Effekt: Name  $\langle id \rangle$  wird an den Wert von  $\langle e \rangle$  gebunden.

Namen können in Scheme fast beliebig gewählt werden, solange:

1. die Zeichen (kommt noch) nicht vorkommen
2. der Name nicht einem numerischen Literal gleicht
3. kein whitespace (Leerzeichen, Tabulatoren, Return) enthalten ist.

Bsp.:  $\text{euro} \rightarrow \text{us\$}$

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

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

$$(\text{lambda } (\langle p1 \rangle \langle p2 \rangle \dots) \langle e \rangle), \quad \langle e \rangle \text{ Rumpf}$$

$\langle e \rangle$  enthält Vorkommen der Parameter  $\langle p1 \rangle, \langle p2 \rangle \dots$

$(\text{lambda } \dots)$  ist eine Spezialform. Wert der Lambda-Abstraktion ist  $\# \langle procedure \rangle$

Anwendung (auch: Applikation/Aufruf) der Lambda-Abstraktion führt zur Ersetzung der vorkommenden Parameter im Rumpf durch die angegebenen Argumente:

(lambda (days) (\* days (\* 155 min-in-a-day)))  $\rightsquigarrow$  (\* 365 (\* 155 min-in-a-day))  $\rightsquigarrow$  81468000

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

Prozeduren sollten im Programm eine ein-bis zweizeiliger Kurzbeschreibung direkt voran gestellt werden.