Fortgeschrittene Programmierung / SS'15

Michael Hanus

9. Juli 2015

Detaillierter Vorlesungsverlauf

14.4. Java Generics: parametrisierte Containerklasse, Typparameter, eingeschränkte Typparameter, Wildcards, beschränkte Wildcards, Typinferenz

16.4. Nebenläufige Programmierung in Java: grundlegende Begriffe, Synchronisationsproblem, Semaphore, Dining Philosophers, Klasse `Thread`, Interface `Runnable`, Eigenschaften von Thread-Objekten

21.4. Synchronisation von Threads in Java, `synchronized`-Methoden, `synchronized`-Anweisung, synchronisierte Methoden vs. synchronisierte Anweisungen, synchronisierte Collections, `wait`, `notify`, `nodifyAll`, Kommunikation zwischen Threads

23.4. sinnvolle Benutzung von `wait`, `notify`, `nodifyAll`, einelementiger Puffer, Benutzung von Synchronisationsobjekten, Beenden und Unterbrechen von Threads, `InterruptedException`, Serialisierung von Daten

28.4. Idee von RMI, Parameterübertragung, Serverseite, Clientseite, RMI-Registrierung, Probleme bei Client-side Synchronisation mit RMI
Einführung in die funktionale Programmierung: Variablenbegriff, Programm, Funktionsdefinitionen, Ausdrücke, Beispiel `square`

30.4.: Beispiele `min`/`fac`, Auswertungsmöglichkeiten, Fibonacci-Zahlen, lokale Definitionen, Layout-Regel, Vorteile lokaler Definitionen

5.5.: Basisdatentypen, Typannotationen, algebraische Datentypen (Aufzählungstypen, Verbundtypen, gemischte Typen, Listen), Operatoren, Vergleich und Ausgabe von Daten (`deriving (Eq,Ord,Show)`)

7.5. Polymorphe Funktionen (`length`, `(++)`, `last`), Definition polymorpher algebraischer Datentypen, `Maybe`, Binärbäume, `String`, `Either`, Tupel (`fst`, `snd`, `zip`)

12.5. `unzip`, Pattern Matching (Patternaufbau, case-Ausdrücke), Guards, Funktionen höherer Ordnung, anynome Funktionen

19.5. partielle Applikation, Currying, Sections, generische Programmierung (`map`, `foldr`, `filter`)

21.5. `foldl`, Kontrollstrukturen als Funktionen höherer Ordnung (`while`), Funktionen als Datenstrukturen (Implementierung von Feldern), wichtige Funktionen höherer Ordnung (Komposition, `flip`, `curry/uncurry`

26.5. Motivation und Struktur von Typklassen vordefinierte Funktionen in Typklassen, Standardklassen, `deriving`, Klasse `Read` und Funktionen `read` und `reads`

28.5. Lazy Evaluation (`from`, `primes`, `fibs`), Sharing, Graphreduktion

2.6. Idee der Ein-/Ausgabe, I/O-Aktionen, do-Notation

4.6. Beispiel Ausgabe von Zwischenergebnissen, list comprehensions, Module

9.6. Einführung in die Logikprogrammierung: Motivation, Verwandtschaftsbeispiel, Prolog-Programme, Fakten, Regeln, Anfragen

11.6. Prolog-Syntax (Zahlen, Atome, Strukturen, Listen), Operatoren, Variablen, Rechnen mit Listenstrukturen, Gleichheit von Termen; Programmiertechnik Aufzählung des Suchraumes (Färben einer Landkarte, Sortieren von Zahlenlisten)

23.6. Programmiertechnik Musterorientierte Wissensrepräsentation (`append`), Verwendung von Relationen, Peano-Zahlen

25.6. Rechnen in der Logikprogrammierung: einfaches Resolutionsprinzip, Substitution, Unifikation, mgu, Unifikationsalgorithmus, occur check, Komplexität

30.6. allgemeines Resolutionsprinzip (SLD-Resolution), Auswertungsstrategie und SLD-Baum Beweisstrategie von Prolog, Endlosschleifen, Negation als Fehlschlag, Probleme der Negation

2.7. Cut-Operator, Fallunterscheidung, Arithmetik in Prolog (`is`, Fakultätsfunktion), arithmetische Constraints, Beispiel Schaltkreisanalyse, Beispiel Hypothekenberechnung

7.7. Constraint-Programmierung über endlichen Bereichen, allgemeines Vorgehen, send-more-money-Beispiel, 8-Damen-Problem, verbesserte Labeling-Strategien

9.7. Weitere FD-Constraints, CP in anderen Sprachen; Meta-Programmierung: Kapselung des Nichtdeterminismus (`findall`, `bagof`, `setof`), Veränderung der Wissensbasis (`assert`, `retract`), Meta-Interpretierer zur Beweislängenberechnung