Fortgeschrittene Programmierung / WS 16/17

Michael Hanus

2. Februar 2017

Detaillierter Vorlesungsverlauf

25.10. Organsiatorisches;
Nebenläufige Programmierung in Java: grundlegende Begriffe, Synchronisationsproblem, Semaphore

27.10. Dining Philosophers, Klasse `Thread`, Interface `Runnable`, Eigenschaften von Thread-Objekten, Monitor-Konzept, Synchronisation von Threads in Java, `synchronized`-Methoden,

1.11. `synchronized`-Anweisung, synchronisierte Methoden vs. synchronisierte Anweisungen, synchronisierte Collections, `wait`, `notify`, `nodifyAll`, Kommunikation zwischen Threads, sinnvolle Benutzung von `wait`, `notify`, `notifyAll`, einelementiger Puffer

3.11. Benutzung von Synchronisationsobjekten, Beenden und Unterbrechen von Threads, `InterruptedException`
Serialisierung von Daten, Idee von RMI, Parameterübertragung, Serverseite, Clientseite, RMI-Registrierung

8.11. Probleme bei Client-side Synchronisation mit RMI
Einführung in die funktionale Programmierung: Variablenbegriff, Programm, Funktionsdefinitionen, Ausdrücke, Beispiel `square`, Beispiele `min`/`fac`, Auswertungsmöglichkeiten, Fibonacci-Zahlen (rekursiv)

10.11.: Fibonacci-Zahlen (iterativ), lokale Definitionen, Layout-Regel, Vorteile lokaler Definitionen; Basisdatentypen, Typannotationen, algebraische Datentypen (Aufzählungstypen, Verbundtypen, gemischte Typen, Listen), Operatoren, Vergleich und Ausgabe von Daten (`deriving (Eq,Ord,Sh`

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

17.11. `unzip`, Pattern Matching (Patternaufbau, case-Ausdrücke), Guards, Funktionen höherer Ordnung, anynome Funktionen, partielle Applikation, Currying, Sections, `flip`

22.11. generische Programmierung (`map`, `foldr`, `filter`, `foldl`), Kontrollstrukturen als Funktionen höherer Ordnung (`while`), Funktionen als Datenstrukturen (Implementierung von Feldern),

24.11. wichtige Funktionen höherer Ordnung (Komposition, `curry/uncurry`, `const`), Funktionen höherer Ordnung in imperativen Sprachen (Ruby, Java 8); Motivation und Struktur von Typklassen

29.11. Instanzen, vordefinierte Funktionen in Typklassen, Standardklassen, `deriving`, Klasse `Read` und Funktionen `read` und `reads`; Unterschiede bei Auswertungsstrategien, Programmsignatur, Terme, Programm, Termersetzungssystem

1.12. Substitution, Position, Teilterm, Reduktionsschritt, Normalform, Reduktionsstrategien (LI, RI, LO, RO, PI, PO), Berechnungsstärke von outermost, Rechnen mit unendlichen Datenstrukturen (`from`, `primes`, `fibs`)

6.12. `repeat`, `iterate`, arithmetische und andere Sequenzen, Lazy Evaluation, Sharing, Graphreduktion; Idee der Ein-/Ausgabe

8.12. I/O-Aktionen, do-Notation, Beispiel Ausgabe von Zwischenergebnissen, I/O-Aktionen zum Lesen und Schreiben von Dateien

13.12. Zeilen einer Datei numerieren, list comprehensions, Module und Exportdeklarationen

15.12. Module: Importdeklarationen; Testen mit QuickCheck: Eigenschaften, `==>`, Referenzimplementierung, Regressionstests, Eingabeklassifikation mit `classify` und `collect`

20.12. Eigene Definitionen von Testdaten: Klasse `Arbitrary`, `elements`, `choose`, `oneof`, `sized`, `vector`, Fallstudien Peano-Arithmetik, `frequency`, Datenabstraktion, rationale Zahlen

22.12. Abstraktion rationaler Zahlen, abstrakter Datentyp, Rat-ADT, Mengen-ADT, unterschiedliche Implementierung und Testen von Mengen

10.1. Einführung in die Logikprogrammierung: Motivation, Verwandtschaftsbeispiel, Prolog-Programme, Fakten, Regeln, Anfragen Prolog-Syntax (Zahlen, Atome, Strukturen, Listen), Variablen, Rechnen mit Listenstrukturen

12.1. Operatoren, Gleichheit von Termen; Programmiertechnik Aufzählung des Suchraumes (Färben einer Landkarte, Sortieren von Zahlenlisten), Programmiertechnik Musterorientierte Wissensrepräsentation (`append`), Verwendung von Relationen, Peano-Zahlen (Definition, Addition, Subtraktion)

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

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

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

31.1. Constraint-Programmierung über endlichen Bereichen, allgemeines Vorgehen, send-more-money-Beispiel, 8-Damen-Problem, verbesserte Labeling-Strategien, weitere FD-Constraints, CP in anderen Sprachen

2.2. Meta-Programmierung: Predikate höherer Ordnung (`call`, `maplist`), Kapselung des Nichtdeterminismus (`findall`, `bagof`, `setof`), Veränderung der Wissensbasis (`assert`, `retract`), Meta-Interpretierer zur Beweislängenberechnung