Fortgeschrittene Programmierung
WS 18/19

Michael Hanus

5. Februar 2019

Detaillierter Vorlesungsverlauf

22.10. Organsiatorisches;
Nebenläufige Programmierung in Java: grundlegende Begriffe, Synchronisationsproblem, Semaphore, Dining Philosophers

23.10. Klasse `Thread`, Interface `Runnable`, Eigenschaften von Thread-Objekten, Monitor-Konzept, Synchronisation von Threads in Java, `synchronized`-Methoden, `synchronized`-Anweisung, synchronisierte Methoden vs. synchronisierte Anweisungen, synchronisierte Collections, `wait`, `notify`, `notifyAll`, Kommunikation zwischen Threads

29.10. sinnvolle Benutzung von `wait`, `notify`, `notifyAll`, einelementiger Puffer, Benutzung von Synchronisationsobjekten, Beenden und Unterbrechen von Threads, `InterruptedException`

30.10. Serialisierung von Daten, Idee von RMI, Parameterübertragung, Serverseite, Clientseite, RMI-Registrierung, Probleme bei Client-side Synchronisation mit RMI

5.11. Einführung in die funktionale Programmierung: Variablenbegriff, Programm, Funktionsdefinitionen, Ausdrücke, Beispiel `square`, Beispiele `min`/`fac`, Auswertungsmöglichkeiten, Fibonacci-Zahlen (rekursiv und iterativ)

6.11.: lokale Definitionen, Layout-Regel, Vorteile lokaler Definitionen; Basisdatentypen, Typannotationen, algebraische Datentypen (Aufzählungstypen, Verbundtypen, gemischte Typen, Listen)

12.11. Operatoren, Ausgabe von Daten (`deriving Show`) polymorphe Funktionen (`length`, `(++)`, `last`), Definition polymorpher algebraischer Datentypen, `Maybe`, Binärbäume, `String`, `Either`

13.11. Tupel (`fst`, `snd`, `zip`, `unzip`), Pattern Matching (Patternaufbau, case-Ausdrücke), Guards, Funktionen höherer Ordnung, anynome Funktionen, partielle Applikation, Currying, Sections, Funktion `flip`,

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

20.11. wichtige Funktionen höherer Ordnung (Komposition, `curry`/`uncurry`, `const`), Funktionen höherer Ordnung in imperativen Sprachen (Ruby, Java 8)

26.11. Motivation und Struktur von Typklassen, Instanzen, vordefinierte Funktionen in Typklassen, Standardklassen, `deriving`, Klasse `Read` und Funktionen `read` und `reads`; Unterschiede bei Auswertungsstrategien, Programmsignatur, Terme

27.11. Programm, Termersetzungssystem, Substitution, Position, Teilterm, Reduktionsschritt, Normalform, Wertaufruf, Namensaufruf, Reduktionsstrategien (LI, RI, LO, RO, PI, PO), Berechnungsstärke von outermost

3.12. Rechnen mit unendlichen Datenstrukturen (`from`, `primes`, `fibs`) `repeat`, `iterate`, arithmetische und andere Sequenzen, Typklassen `Enum` und `Bounded`, Lazy Evaluation, Sharing, Graphreduktion

4.12. List comprehensions, Anwendung CSV read/show, Idee der Ein-/Ausgabe, I/O-Aktionen

10.12. Aktionen zur Ein- und Ausgabe, do-Notation, Beispiel Ausgabe von Zwischenergebnissen, I/O-Aktionen zum Lesen und Schreiben von Dateien, Zeilen einer Datei numerieren,

11.12. Module, Exportdeklarationen, Importdeklarationen; Transformationen auf Containerstrukturen: `Functor`, `fmap`, Transformationen mit beliebigen Funktionen: `Applicative`, `pure`, `<*>`, arithmetische Ausdrücke und deren Auswertung

17.12. Arithmetische Ausdrücke und deren Auswertung, `Applicative`-Gesetze, `Applicative`-Instanzen für Listen und `IO`, effektvolle Berechnungen, Verbesserung der Auswertung arithmetischeer Ausdrücke durch monadische Struktur, Klasse `Monad`, `Monad`-Instanz für Listen; Einführung in das Testen mit QuickCheck

18.12. Testen mit QuickCheck: Eigenschaften, `==>`, Referenzimplementierung, Regressionstests, Eingabeklassifikation mit `classify` und `collect`, eigene Definitionen von Testdaten: Klasse `Arbitrary`, `elements`, `choose`, `oneof`, `sized`, `vector`, Fallstudien Peano-Arithmetik, `frequency`

7.1. Datenabstraktion, rationale Zahlen, Abstraktion rationaler Zahlen, abstrakter Datentyp, Rat-ADT, Mengen-ADT, Implementierung und Testen von Mengen

8.1. Implementierung von Mengen als ungeordnete und geordnete Listen

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

15.1. Prolog-Syntax (Zahlen, Atome, Strukturen, Listen), Variablen, Rechnen mit Listenstrukturen Operatoren, Gleichheit von Termen; Programmiertechnik Aufzählung des Suchraumes (Sortieren von Zahlenlisten), Programmiertechnik Musterorientierte Wissensrepräsentation (`append`), Verwendung von Relationen

21.1. Peano-Zahlen (Definition, Addition, Subtraktion), Rechnen in der Logikprogrammierung: einfaches Resolutionsprinzip, Substitution, Unifikation, mgu, Unifikationsalgorithmus, occur check

22.1. Unifikationsalgorithmus: Komplexität, allgemeines Resolutionsprinzip (SLD-Resolution), Auswertungsstrategie und SLD-Baum, Beweisstrategie von Prolog, Endlosschleifen

28.1. Negation als Fehlschlag, Probleme der Negation, verzögerte Negation mit Coroutining; Cut-Operator, Fallunterscheidung, Arithmetik in Prolog (`is`, Fakultätsfunktion), arithmetische Constraints, Beispiel Schaltkreisanalyse

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

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

5.2. Prädikate zur Ein- und Ausgabe von Daten, Tracing von Prolog-Programmen; Kurzüberblick zu Multiparadigmen-Sprachen, Einführung in Curry (Verwandtschaftsbeispiel), funktionale Muster (`last`, `perm`), bedarfsgesteuerte Suche in Curry