Fortgeschrittene Programmierung
WS 19/20

Michael Hanus

28. Januar 2020

Detaillierter Vorlesungsverlauf

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

22.10. 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, Kommunikation zwischen Threads, `wait`, `notify`, `notifyAll`

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

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

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

5.11.: Lokale Definitionen, Layout-Regel, Vorteile lokaler Definitionen; Basisdatentypen, Typannotationen, algebraische Datentypen (Aufzählungstypen, Verbundtypen, gemischte Typen, Listen), Ausgabe von Daten (`deriving Show`), Operatoren

11.11. polymorphe Funktionen (`length`, `(++)`, `last`), Definition polymorpher algebraischer Datentypen, `Maybe` und `maybeLast`, Binärbäume, `String`, `Either`, Tupel (`fst`)

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

18.11. Sections, Funktion `flip`, generische Programmierung (`map`, `foldr`, `filter`, `foldl`), Kontrollstrukturen als Funktionen höherer Ordnung (`while`)

19.11. Funktionen als Datenstrukturen (Implementierung von Feldern), wichtige Funktionen höherer Ordnung (Komposition, `curry`/`uncurry`, `const`), Funktionen höherer Ordnung in imperativen Sprachen (Ruby, Java 8, JavaScript)

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

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

2.12. Rechnen mit unendlichen Datenstrukturen (`fibs`, `repeat`, `iterate`), arithmetische und andere Sequenzen, Typklassen `Enum` und `Bounded`, Lazy Evaluation, Sharing, Graphreduktion, List comprehensions

3.12. Anwendung CSV read/show, Idee der Ein-/Ausgabe, I/O-Aktionen, Aktionen zur Ein- und Ausgabe, do-Notation, Beispiel Ausgabe von Zwischenergebnissen

9.12. I/O-Aktionen zum Lesen und Schreiben von Dateien, Zeilen einer Datei numerieren, Module, Exportdeklarationen, Importdeklarationen; Transformationen auf Containerstrukturen: `Functor`, `fmap`

10.12. `Functor`-Gesetze, Transformationen mit beliebigen Funktionen: `Applicative`, `pure`, `<*>`, arithmetische Ausdrücke und deren Auswertung, `Applicative`-Gesetze, `Applicative`-Instanzen für Listen und `IO`, effektvolle Berechnungen, Verbesserung der Auswertung arithmetischer Ausdrücke durch monadische Struktur, Klasse `Monad`,

16.12. `Monad`-Instanz für Listen; Testen mit QuickCheck: Eigenschaften, `==>`, Referenzimplementierung, Regressionstests, Eingabeklassifikation mit `classify` und `collect`, eigene Definitionen von Testdaten: Klasse `Arbitrary`, `elements`, `choose`, `oneof`, `sized`, `vector`,

17.12. Fallstudien Peano-Arithmetik, `frequency`, Datenabstraktion, rationale Zahlen, Abstraktion rationaler Zahlen, abstrakter Datentyp, Rat-ADT

7.1. Mengen-ADT, Implementierung und Testen von Mengen, Implementierung von Mengen als ungeordnete und geordnete Listen

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

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

14.1. Programmiertechnik Musterorientierte Wissensrepräsentation (`append`), Verwendung von Relationen, Peano-Zahlen (Definition, Addition, Subtraktion), Rechnen in der Logikprogrammierung: einfaches Resolutionsprinzip, Substitution, Unifikator, mgu, disagreement sets

20.1. Unifikationsalgorithmus, occur check, Komplexität, allgemeines Resolutionsprinzip (SLD-Resolution), Auswertungsstrategie und SLD-Baum, Beweisstrategie von Prolog, Endlosschleifen

21.1. Negation als Fehlschlag, Probleme der Negation, Cut-Operator, Fallunterscheidung, Arithmetik in Prolog (`is`, Fakultätsfunktion), arithmetische Constraints, Beispiel Hypothekenberechnung, Constraint-Programmierung über endlichen Bereichen, allgemeines Vorgehen, send-more-money-Beispiel

27.1. 8-Damen-Problem, verbesserte Labeling-Strategien, Meta-Programmierung: Prädikate höherer Ordnung (`call`, `maplist`), Kapselung des Nichtdeterminismus (`findall`, `bagof`, `setof`), Veränderung der Wissensbasis (`assert`, `retract`)

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