Fortgeschrittene Programmierung
WS 21/22

Michael Hanus

1. Februar 2022

Detaillierter Vorlesungsverlauf

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

26.10. 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`

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

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

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

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

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

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

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

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

29.11. Typklassen für polymorphe Datentypen, Standardklassen, `deriving`, Klasse `Read` und Funktionen `read` und `reads`; Unterschiede bei Auswertungsstrategien, Programmsignatur, Terme

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

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

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

13.12. Zeilen einer Datei numerieren, Module, Exportdeklarationen, Importdeklarationen; Transformationen auf Containerstrukturen: `Functor`, `fmap`, Transformationen mit beliebigen Funktionen: `pure`, `<*>`

14.12. `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`, `Monad`-Instanz für Listen; Einführung zu eigenschaftsbasierten Testen

20.12. 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`

21.12. Datenabstraktion, rationale Zahlen, Abstraktion rationaler Zahlen, abstrakter Datentyp, Rat-ADT, Mengen-ADT, Implementierung und Testen von Mengen, Implementierung von Mengen als ungeordnete Listen, Implementierung von Mengen als geordnete Listen

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

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

17.1. Rechnen in der Logikprogrammierung: einfaches Resolutionsprinzip, Substitution, Unifikator, mgu, disagreement sets, Unifikationsalgorithmus, occur check, Komplexität

18.1. allgemeines Resolutionsprinzip (SLD-Resolution), Auswertungsstrategie und SLD-Baum, Beweisstrategie von Prolog, Endlosschleifen

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

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

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

1.2. Tracing von Prolog-Programmen, Differenzlisten, Definite Clause Grammars, Kurzüberblick zu Multiparadigmen-Sprachen, Einführung in Curry (Verwandtschaftsbeispiel), funktionale Muster (`last`, `perm`), bedarfsgesteuerte Suche in Curry