Deklarative Programmiersprachen SS 2019

Michael Hanus

2. Juli 2019

Detaillierter Vorlesungsverlauf

8.4.: Einführung: Entwicklung von Sprachkonzepten, referenzielle Transparenz, Vorteile im Vergleich zu imperativen Sprachen (Optimierung, Parallelisierung, Zuverlässigkeit, Lesbarkeit), Beispiel Quicksort, Klassifikation von Programmiersprachen, Anwendungen deklarativer Programmiersprachen, Beispiele: Adresssuche in HTML-Seiten, Su Doku-Löser mit Web-Interface

9.4.: Funktionale Programmierung: Ausdrücke, Funktionsdefinitionen, Auswertung von Ausdrücken, Redex, strikte und nicht-strikte Sprachen, Fallunterscheidung, bedingte Regeln, Muster, Spezifikationssprache vs. deklarative Programmiersprache, Newtonsches Approximationsverfahren, Gültigkeitsbereich, lokale Deklarationen, Layout-Regel; strenge Typisierung, Datentypen, Basistypen strukturierte Typen (Listen, Zeichenketten, Tupel), funktionale Typen, benutzerdefinierte Datentypen

15.4.: *Pattern Matching*: Vorteile von Definitionen mit Pattern Matching, Auswertungsverhalten bei Mustern (Konjunktion, Paralleles Oder), `case`-Ausdrücke, Übersetzung muster-orientierter Funktionsdefinitionen in `case`-Ausdrücke

16.4.: Übersetzung des *parallel or*, Problem überlappender Regeln, Funktion `diag`, prinzipielles Problem sequentieller Auswertungsstrategien, uniforme Funktionsdefinitionen, Reihenfolgeunabhängigkeit
Einführung Typsysteme: streng getypte Sprache, Typ, Typfehler, schwach getypte Sprache

23.4.: *Typsystem:* ad-hoc Polymorphismus, parametrischer Polymorphismus, Typinferenz, typkorrekt, Typausdrücke, (Typ-)Substitution, Typinstanz, Typschema, generische Instanz, Typannahme, Inferenzsystem zur Typprüfung, Typisierung von `twice`, allgemeinster Typ

29.4.: Typprüfung für mehrere Funktionen, Typinferenz, Vorgehen, Berechnung allgemeinster Unifikatoren nach Martelli/Montanari, Typinferenz für mehrere Funktionen

30.4.: statischer Aufrufgraph, Grenzen der Typisierung: Selbstanwendung, `funsum`, Polymorphismus 2. Ordnung
Lazy Evaluation: Redex, Normalform, LO-Reduktion, schwache Kopfnormalform

6.5.: *Reduktionssysteme:* Reduktionssystem, Reduktionsrelationen, reduzierbar, irreduzibel, Normalform, Church-Rosser, konfluent, lokal konfluent, terminierend, Noethersch, Newman-Lemma, Motivation Termersetzungssysteme (Gruppenaxiome), Signatur, Sorte, Funktionssymbol, Variable, Term, Grundterm, linearer Term, Termersetzungssystem

7.5.: Substitution, Position, Teilterm, Ersetzung, Vergleich von Termpositionen, Reduktionsschritt, Redex kritisches Paar, Kritisches-Paar-Lemma, (schwach) orthogonale Termersetzungssysteme, Reduktionsstrategie (sequenzielle, normalisierende)

13.5.: Reduktionsstrategien LI/LO/PO, linksnormale Termersetzungssysteme Konstruktor, konstruktorbasiertes Termersetzungssystem, definierender Baum, induktiv-sequenziell, Reduktionsstrategie φ, vollständig definierte Funktionen, Eigenschaften von φ

14.5.: Motivation zur logisch-funktionalen Programmierung: Verwandschaftbeispiel, Aspekte der logisch-funktionalen Programmierung Variablen in initialen Ausdrücken, Extravariablen in Regeln, `failed`, nichtdeterministische Operationen ("`?`"), Logiksprachen als Spezialfall logisch-funktionaler Sprachen, Spiel 24

20.5.: reguläre Ausdrücke (Darstellung, Semantik, Matching und `grep`), Definition von Narrowing, Narrowing als Verallgemeinerung von Reduktion, Gleichungen, gültige Gleichungen, Lösungen von Gleichungen, Korrektheit und Vollständigkeit von Narrowing (Satz von Hullot)

27.5.: Vollständigkeit bzgl. normalisierter Substitutionen, reflexive Gleichheit und deren konstruktive Definition; Logikprogrammierung (Prolog) als Spezialfall der logisch-funktionalen Programmierung, Extravariablen, bedingte Regeln (Transformation in unbedingte), Resolution als Spezialfall von Narrowing, Backtracking, Färben einer Landkarte, Graph zeichnen (Haus vom Nikolaus)

28.5.: Strikte Narrowing-Strategien: Innermost Narrowing, Vollständigkeit, Innermost Basic Narrowing, Rückweisung (rejection), Normalisierendes Innermost Narrowing

3.6.: Lazy Narrowing-Strategien: Outermost Narrowing, Lazy Narrowing, Nachteile, Idee von Needed Narrowing, Steuerung von Needed Narrowing durch definierende Bäume, formale Definition von Needed Narrowing, Eigenschaften (Vollständigkeit, Optimalität, Determinismus)

4.6.: Erweiterte definierende Bäume für überlappende Regeln, Definition und Eigenschaften von Weakly Needed Narrowing; Nichtdeterministische Operationen: Realisierung, intuitive Bedeutung, Anwendung (permutation sort), operationale Vorteile, semantische Probleme (call-time choice vs. run-time choice)
Residuation: operationale Idee, Vorteile

11.6.: Anschluss externer Funktionen mittels Residuation, Nachteile von Residuation (Unvollständigkeit, unendliche Berechnungen mit verzögerten Constraints), Kombination von Residuation und Narrowing (flex/rigid branches), Strategie für diese Kombination, Auswertungsstrategie von Curry (optimale definierende Bäume), Gleichheitstest, Gleichheitsconstraint

17.6.: bedingte Regeln, Funktionen höherer Ordnung (rigides `apply`), Modellierung nebenläufiger Objekte (Bankkonto), Erweiterungen: Constraint-Programmierung, CLP(R) (Hypothekenberechnung in Curry), CLP(FD): send-more-money-Beispiel, vordefinierte FD-Constraints, SuDoku-Löser

18.6.: Eingekapselte Suche: Motivation, `findall`, Probleme, starke und schwache Einkapselung,

Probleme der starken und schwachen Einkapselung, rigide Einkapselung für logische Variablen, set functions, Datentyp `Values` und seine Operationen

24.6.: Beispiel kürzeste Wegesuche Beispiel 8-Damen-Problem mit Set Functions, Funktionale Muster: Probleme der strikten Gleichheit bei `last`, Definition von `last` mit funktionalen Muster, deklarative Bedeutung funktionaler Muster, Ebenenabbildung und stratifizierte Programme, Unifikation "`=:<=`" für funktionale Muster

25.6.: Beispiele `perm`, `descending`, `sort`, `lengthUpToRepeat`, Vereinfachung arithmetischer Ausdrücke
GUI-Programmierung: Widgets, Layout, Event Handler, logische Variablen als Referenzen, Zähler-GUI

1.7.: Kompositionalität von GUIs (vier Zähler-GUIs), Temperaturkonverter-GUI, Tischrechner-GUI, IO-Referenzen, Analyse von Zugangsprotokollen mit Curry, XML als semistrukturiertes Austauschformat

2.7.: Darstellung von XML-Dokumenten, Pattern Matching auf XML-Dokumenten, partielle Muster, ungeordnete Muster, tiefe Muster, negierte Muster, Transformation von XML-Dokumenten
Zusammenfassung