Deklarative Programmiersprachen SS 2024

Michael Hanus

8. Juli 2024

Detaillierter Vorlesungsverlauf

15.4.: Einführung: Entwicklung von Sprachkonzepten, referenzielle Transparenz, Vorteile im Vergleich zu imperativen Sprachen (Optimierung, Parallelisierung, Zuverlässigkeit, Lesbarkeit), Beispiel Quicksort, Klassifikation von Programmiersprachen, Anwendungen deklarativer Programmiersprachen,

16.4.: 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)

18.4.: funktionale Typen, Curryfizierung, benutzerdefinierte Datentypen, *Pattern Matching*: Vorteile von Definitionen mit Pattern Matching, Auswertungsverhalten bei Mustern (Konjunktion, Paralleles Oder), `case`-Ausdrücke

22.4.: Übersetzung muster-orientierter Funktionsdefinitionen in `case`-Ausdrücke, Übersetzung des *parallel or*, Problem überlappender Regeln, Funktion `diag`, prinzipielles Problem sequentieller Auswertungsstrategien, uniforme Funktionsdefinitionen, Reihenfolgeunabhängigkeit

23.4.: *Typsysteme:* streng getypte Sprache, Typ, Typfehler, schwach getypte Sprache, ad-hoc Polymorphismus, parametrischer Polymorphismus, Typinferenz, typkorrekt, Typausdrücke, (Typ-)Substitution, Typinstanz, Typschema, generische Instanz, Typannahme, Inferenzsystem zur Typprüfung

25.4.: Typisierung von `twice`, allgemeinster Typ, Typprüfung für mehrere Funktionen, Typinferenz, Vorgehen, Berechnung allgemeinster Unifikatoren nach Martelli/Montanari

6.5.: Typinferenz für mehrere Funktionen, statischer Aufrufgraph, Grenzen der Typisierung: Selbstanwendung, `funsum`, Polymorphismus 2. Ordnung
Lazy Evaluation: Redex, Normalform, LO-Reduktion, schwache Kopfnormalform

Reduktionssysteme: Reduktionssystem, Reduktionsrelationen, reduzierbar, irreduzibel, Normalform, Church-Rosser, konfluent, lokal konfluent

7.5.: terminierend, Noethersch, Newman-Lemma, Motivation Termersetzungssysteme (Gruppenaxiome), Signatur, Sorte, Funktionssymbol, Variable, Term, Grundterm, linearer Term, Termersetzungssystem, Substitution, Position, Teilterm, Ersetzung, Vergleich von Termpositionen, Reduktionsschritt, kritisches Paar, Kritisches-Paar-Lemma, (schwach) orthogonale Termersetzungssysteme

13.5.: Reduktionsstrategie (sequenzielle, normalisierende), Reduktionsstrategien LI/LO/PO, linksnormale Termersetzungssysteme

14.5.: Konstruktor, konstruktorbasiertes Termersetzungssystem, definierender Baum, induktiv-sequenziell, Reduktionsstrategie φ, vollständig definierte Funktionen, Eigenschaften von φ

21.5.: 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

27.5.: 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)

28.5.: 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)

3.6.: Strikte Narrowing-Strategien: Innermost Narrowing, Vollständigkeit, Innermost Basic Narrowing, Rückweisung (rejection), Normalisierendes Innermost Narrowing Lazy Narrowing-Strategien: Outermost Narrowing, Idee von Lazy Narrowing

4.6.: Lazy Narrowing, Nachteile, Idee von Needed Narrowing, Steuerung von Needed Narrowing durch definierende Bäume, Eigenschaften (Vollständigkeit, Optimalität, Determinismus), erweiterte definierende Bäume für überlappende Regeln, Definition und Eigenschaften von Weakly Needed Narrowing; Nichtdeterministische Operationen: Realisierung, intuitive Bedeutung, Anwendung (permutation sort), operationale Vorteile

10.6.: Semantische Probleme bei nichtdeterministischen Argumenten (call-time choice vs. run-time choice); Residuation: operationale Idee, Vorteile, 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) bedingte Regeln, Funktionen höherer Ordnung (rigides `apply`)

11.6.: Gleichheitstest, strikte Gleichheit, Unterschied `Eq` und `Data`, Typ freier Variablen, Gleichheitsconstraint, Unifikation, Modellierung nebenläufiger Objekte (Bankkonto)

17.6.: Eingekapselte Suche: Motivation, `allSolutions`, Probleme, starke und schwache Einkapselung, Probleme der starken und schwachen Einkapselung, rigide Einkapselung für logische Variablen, set functions, Datentyp `Values` und seine Operationen

18.6.: 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

24.6.: Beispiele `perm`, `descending`, `sort`, `lengthUpToRepeat`, Vereinfachung arithmetischer Ausdrücke; Anwendung GUI-Programmierung: Widgets, Layout

25.6.: GUI-Programmierung: Widgets mit Event Handler, logische Variablen als Referenzen, Zähler-GUI, Kompositionalität von GUIs (vier Zähler-GUIs), Temperaturkonverter-GUI, Tischrechner-GUI, IO-Referenzen, Web-Programmierung mit Curry (Beispiel `RevDup`)

1.7.: Analyse von Zugangsprotokollen mit Curry, XML als semi-strukturiertes Austauschformat, Darstellung von XML-Dokumenten

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

8.7.: Default-Regeln in Curry, deterministische Operationen, Vor- und Nachbedingungen in Curry-Programmen, Verifikation nichtfehlschlagender Programme; Zusammenfassung