Deklarative Programmiersprachen SS 2024

Michael Hanus

25. April 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