Besondere Funktionsverwendungen deklarativ-funktionaler Sprachen

1 Haskell – Funktionen als algebraische Objekte

1.1 Funktionen mit Gesetzen

In Haskell sind Funktionen algebraischen Strukturen unterworfen: Functor, Applicative, Monad, Semigroup, Monoid, Category, Arrow. Man programmiert keine Abläufe, sondern Gesetzssysteme.

1.2 Funktionen über Typkonstruktoren

Funktionen operieren über ganze Typfamilien:

$$f : \text{Functor } f \Rightarrow f\,a \to f\,b$$

1.3 Komposition statt Kontrolle

$$result = transform \circ validate \circ parse$$

2 Idris 2 – Funktionen als Beweise

2.1 Abhängige Typen

$$append : \text{Vect } n\,a \to \text{Vect } m\,a \to \text{Vect } (n+m)\,a$$

2.2 Programme als logische Aussagen

$$plusZeroRightNeutral : (n : Nat) \to n + 0 = n$$

2.3 Totalität

Alle Fälle müssen abgedeckt sein, Terminierung ist garantiert.

3 Lisp – Funktionen als formbare Syntax

3.1 Code als Daten

Programme sind Datenstrukturen.

3.2 Makros

Makros transformieren abstrakte Syntaxbäume.

4 Scala – Funktionen als Integrationswerkzeug

4.1 Effekte als Werte

Seiteneffekte werden beschrieben, nicht ausgeführt.

5 Gemeinsamer Kern

Funktionen beschreiben Wahrheit, nicht Ablauf.