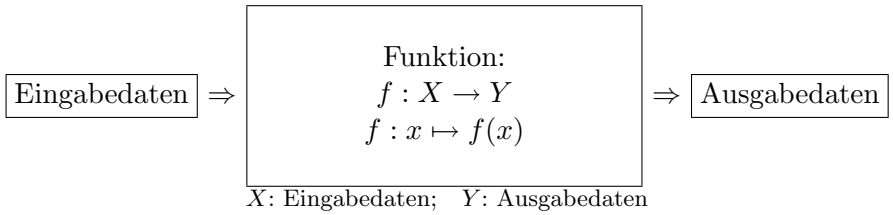
Funktionale Programmierung

Jan Krieger

28. November 2003

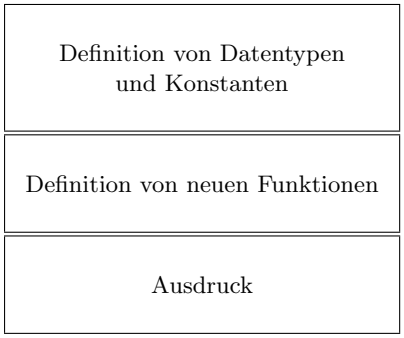
- **Programm** = Abbildung von Eingabedaten auf Ausgabedaten



→ Übertragen des mathematischen Konstruktes einer Abbildung auf die Programmierung, z.B.:

```
sin(0)  -- gueltiges Programm
5+3*2   -- gueltiges Programm
```

→ zusätzlich in funktionalen Programmen: Definition von Datentypen und Konstanten, sowie Definition von neuen Funktionen (alles optional).



→ Funktionen $f : X \to Y \quad f : x \mapsto f(x)$ mit $x \in X; \quad f(x) \in Y$
X, Y können belibige Mengen sein, z.B. auch karthesisches Produkt (z.B. $X = \mathbb{R} \times \mathbb{R}$, dann $x \in X \Rightarrow x = (a, b)$ 2-er Tupel)
z.B.: Logarithmusfunktion:

$$log : (\mathbb{R} \times \mathbb{R}) \to \mathbb{R}$$

$$log : (a, b) \mapsto \log_b a$$

- **Datentypen:** geben an, welcher Art ein Argument, oder der Rückgabewert einer Funktion ist. In der Mathematik z.B.: $\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{C}, \mathbb{R} \times \mathbb{R} = \mathbb{R}^2, \mathbb{R}^3$, oder Mengen $B = \{1, 0\}, X = \{rot, blau, gelb\}$
  In der Informatik gibt es Datentypen, die mehr oder weniger den obigen Mengen entsprechen. Im folgende sind (einige) Datentypen von gofer aufgeführt:

`Int` $\hat{=}\mathbb{Z}$; `Float` $\hat{=}\mathbb{R}$; `Bool` $\hat{=}\{true, false\}$;
Allerdings sind die Wertebereiche beschrnkt, also nicht alle Zahlen darstellbar (bei `Float` wird gerundet, `Int` umfasst z.B. den Bereich von $-32767..32768$)
`Char` fasst ein Zeichen. Einzelne Zeichen werden in einfache Anführungsstrich gesetzt: `'a'`, `'b'`, ...
`String` sind Zeichenkette: `"Hello World"`
Listen: Beispiel: `['a', 'b', 'c']`, oder die leere Liste `[]`
Anfügen von Elementen an eine Liste (nur am Listenanfang möglich): `'c':['a', 'b']`⇒`['c', 'a', 'b']`
Anfügen von ganzen Listen: `[1, 2] ++ [3, 2]` ⇒ `[1, 2, 3, 2]`
selbstdefinierte Datentypen: Beispiel:

```
data Temperatur Fahrenheit Float | Celsius Float
```

data: Schlüsselwort, Temperatur: Name des neuen Datentypes, neuer Datentyp ist entweder Fahrenheit-Temperatur, oder Celsius-Temperatur, wobei das Wort Fahrenheit bzw. Celsius zum Datentyp gehört. Gültige Werte wären etwa: `Celsius 32.0` oder `Fahrenheit 90.5`. Der Wert `45.7` wäre ungültig, weil Fahrenheit oder Celsius fehlen.

- **Funktionsdefinitionen:** besteht aus:
  Deklaration: `f:: Int -> Int` (lies: die Funktion f (Name) bildet einen Eingabewert vom Typ Int auf einen Ausgabeweet vom Typ Int ab)
  Implementation: `f(x) =x*x` (lies: die Funktion f wird mit einem Parameter x aufgerufen. Ihr Rückgabewert isT (`"="`: Rückgabeoperator) der Ausdruck `x*x`) Funktionen mit mehreren Parameter:

```
f:: (Int, Int) -> Int
f(x, y) = x-2*y
```

  gültige *Ausdrücke* sind:
  Int: `5+2` `23*(6+2)`
  Float: `5.0+3.1415*3`
  Bool: `true && false`, `(a>b)`, Beispiel-Funktion dazu (gibt true, wenn erster Parameter größer als zweiter):

```
groesser:: (Int, Int) -> Bool
groesser(a,b) =(a>b)
```

  In Ausdrücken sind (je nach Typ) verschiedene Operatoren möglich. Hier einige:
  `+ - * / ^(Potenz) mod (Modulo-Division) &&(and) ==(gleich) >=(groesser gleich)`
  usw. siehe auch gofer-Tutorial
  Ein solcher Ausdruck steht auch im gofer-Programm (siehe Anfang des Dokumentes). Er kann natürlich auch Funktionen enthalten und aufrufen!

- **Kontrollstrukturen:** Kontrollstrukturen ermöglichen es, je nach Eingabedaten zu entscheiden, was gemacht werden soll. z.B. verschiedenen Versionen der Fibonacci-Zahlen:

  math. Definition: $fib : \mathbb{N} \to \mathbb{N} fib(n) = \begin{cases} 1 & x = 0 \\ 1 & x = 1 \\ fib(n-2) + fib(n-1) & sonst \end{cases}$

  gofer-Programme dazu (siehe auch Skript !!!):

```
fib:: Int -> Int
```

```
-- erste Implementationsmoeglichkeit mit | (nahe an math. Schreibweise)
fib(n) |n==0       =1
       |n==1       =1
       |otherwise  =fib(n-2)+fib(n-1)

-- zweite Implementationsmoeglichkeit mittel if
fib1(n) =if((n==0) || (n==1^)) then 1 else fib1(n-1)+fib1(n-2)

-- dritte Implementationsmoeglichkeit mittels Pattern-Matching
fib2(0) =1
fib2(1) =1
fib(n+2) = fib(n)+fib(n+1)
```

Hier noch ein paar Informatik-Bücher, die ich ganz gut finde. Viele sollten in der UB stehen (vor Allem die über Algorithmen ...):

[1:] **"Computer-Kurzweil 2."** Spektrum der Wissenschaft Verlagsgesellschaft mbH, Heidelberg 1992

[2:] *Cuber, U. Haselier, R. G. (Hrsg.), Fahnenstrich, K. (Hrsg.)* : **"C-Programmierung. Grundlagen und fortgeschrittene Programmiertechniken.",** ECON Verlag, Düsseldorf München 1997 [2]

[3:] *Herrmann, D.* : **"C++ für Naturwissenschaftler. Beispielorientierte Einführung.",** Addison-Wesley, Bonn München Reading, Mass. 2001, S. 183-204

[4:] *Heun, V.* : **"Grundlegende Algorithmen. Einführung in den Entwurf und die Analyse effizienter Algorithmen.",** Springer, New York Berlin Heidelberg 2000

[5:] *Pepper, P.* : **"Funktionale Programmierung. in OPAL, ML, HASKELL und GOFER.",** Springer, New York Berlin Heidelberg 1998

[6:] *Prinz, P., Kirch-Prinz, U.* : **"C: kurz & gut.",** O'Reilly, Köln Paris Cambridge 2002

[7:] *Sedgewick, R.* : **"Algorithmen.",** Addison Wesley, München Bonn 1991 [3]

[8:] *Stroustrup, B.* : **"Die C++-Programmiersprache.",** Addison-Wesley, Bonn München Reading, Mass. 2000 [4]