FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Einführung FuPS

Funktionale Programmiersprachen (FuPS)
Was ist das?

- Programmierung mit Funktionen

```
def square(x:Int) : Int = x * x
 square(5) -> 25
```

- Rekursion als einzige Kontrollstruktur

```
def fak(x:Int) : Int = if (x==0) 1 else x*fak(x-1)
```

- Funktionen höherer Ordnung

statt:

```
int sumOfOdds(List<Integer> list) {
            int sum = 0;
            for (Integer i : list) {
            if (i%2 != 0)  sum += i;
            }
            return sum;
      }
```

```
def sumOfOdds(xs:List[Int]) : Int
      = fold(filter(xs,_%2 != 0),(x,y) => x+y,0)
```

Warum FuPS?

- besser strukturierte Programme
- keine Variablen
  -> keine Seiteneffekte:
  -> Funktionswerte bei gleichen Parameterwerten eindeutig
- parallele Auswertung der Parameter
- Konzepte in vielen modernen Programmiersprachen vorhanden
  (z.B. Lambda-Expressions ab Java 8.0)
- weniger Fehler (Speicherverwaltung durch Compiler)
  -> zuverlässigere Programme
- eignen sich besser zur Verifikation (sicherheitskritische Systeme)
- Programmentwicklung schneller und einfacher
  Ericsson durch ERLANG 1 10-25 mal schnellere Entwicklungszeit
- Programme 2-10mal kürzer
  -> ideal zur Prototypentwicklung
- besser wiederzuverwenden und modularer strukturiert

Einsatz von FuPS in Industrie
(beruht auf M. Hanus, Uni Kiel)

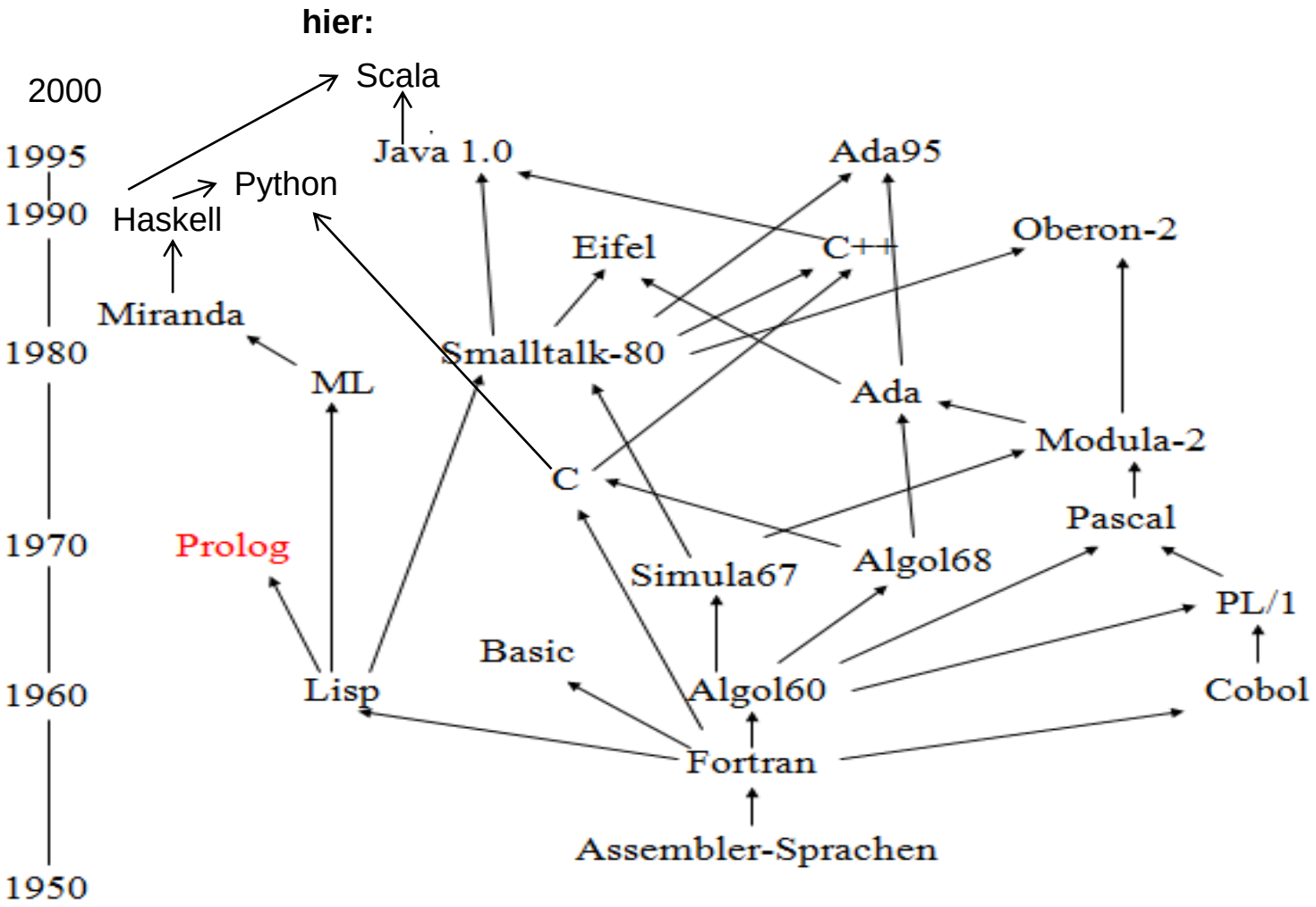
Jane Street Capital, eine Finanzhandelsfirma -> OCaml (Blog).

Galois -> funktionale Programmiersprachen und -konzepte zur Entwicklung sicherheitskritischer Systeme.

Haskell-Anwendungen bei Capital Match
Credit Suisse
Facebook
Twitter
Naughty Dog Inc.
Trifork
Citrix
S&P Capital IQ
Verizon

Forscher bei Microsoft fordern in Zeitschrift CACM, dass Informatik-studierende so früh wie möglich FuPS erlernen sollten.

FuPS im Sprachenkontext



Inhalt der Vorlesung

- Einführung Scala
- Typen, Ausdrücke & Funktionen
- First Order Funktionen
- Listen, Pattern Matching, Case Classes & Objects
- Tail Rekursion
- Auswertung
- Higher Order Funktionen
- Currying & Partielle Application
- Polymorphie & Aufzählungstypen
- Objekt-Funktional, Mutable & Immutable
- Lazy Evaluation
- Lambda Kalkül & Ausdrücke
- Design Pattern
- ….