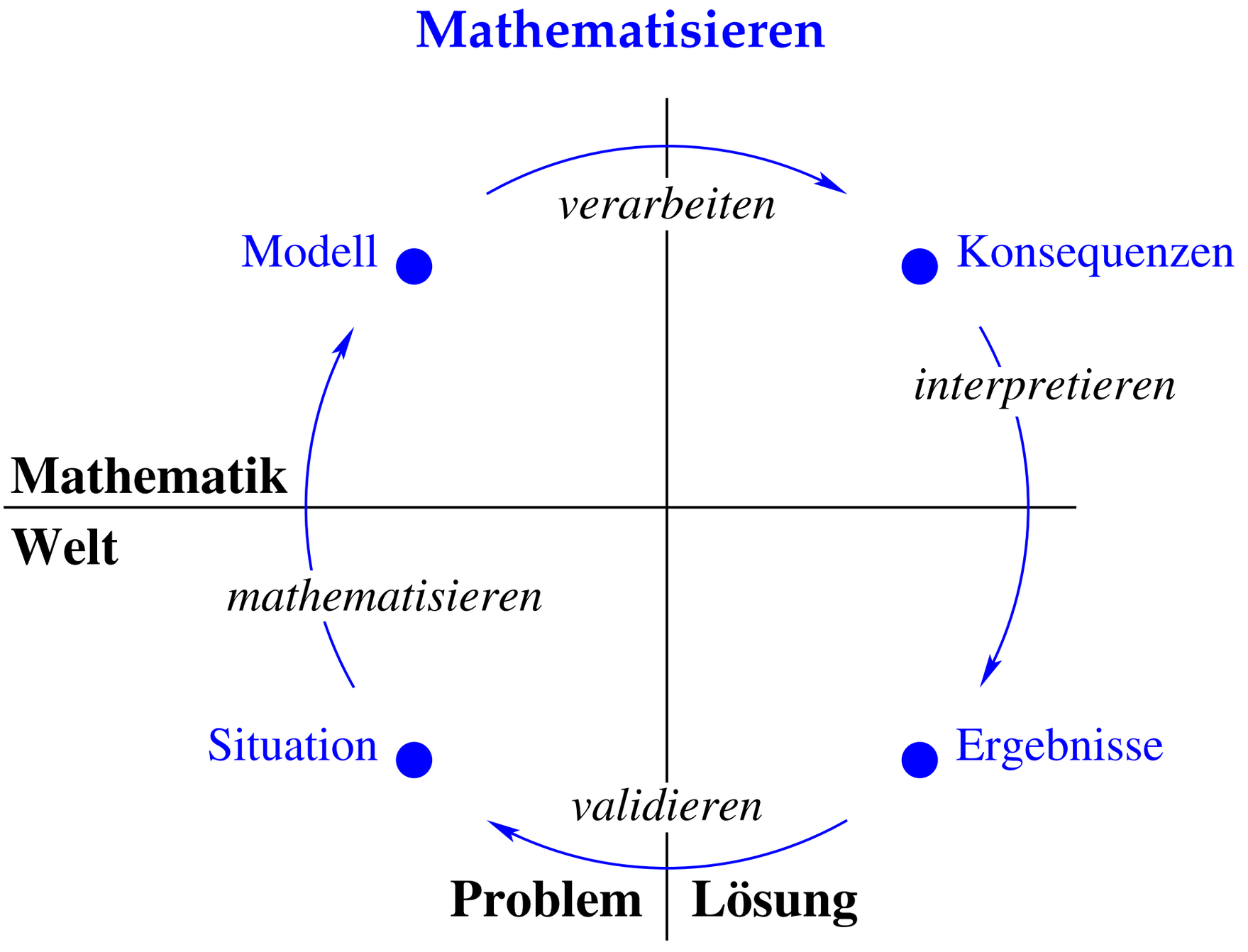
Programmieren als Grundkompetenz

Irrtümer, Chancen und Methodik der Programmierausbildung

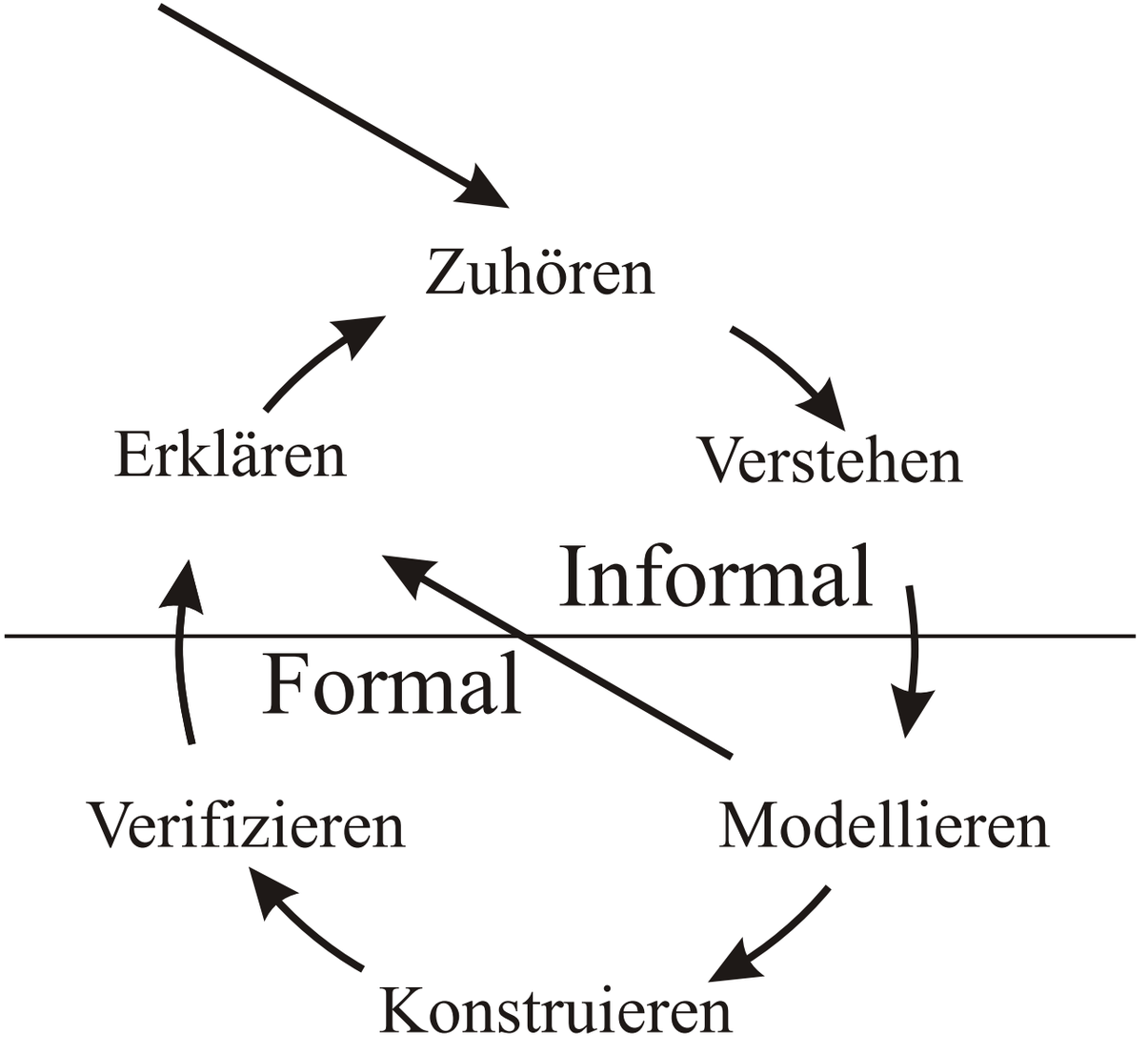
Michael Sperber

Vorgeschichte

- ...
- „AP Computer Science“ (1987/1998)
- „Informatik I“ (viele Male)
- ...
- „Programmieren für Geisteswissenschaftler“ (1999)
- „Informatik I“ (1999/2000, 2000/2001)
- „Programmieren für Geisteswissenschaftler“ (2001/2002)
- **PISA**-Studie (2002)


Mathematisieren
verarbeiten
Modell
Konsequenzen
interpretieren
Mathematik
Welt
mathematisieren
Situation
Ergebnisse
validieren
Problem
Lösung

Programmieren als Form des Problemlösens



Mathematik und Programmieren

PISA-Studie:

Mathematik-Kompetenz
→ *Mathematical Literacy*
→ Problemlösungskompetenz

Techniken:

- Modellierung
- Zerlegung zusammengesetzter Probleme

Programmierung:

- Modellierung
- „Abstraktion, Abstraktion, Abstraktion“
  (Grundtechnik zur Zerlegung zusammengesetzter Probleme)

Programmieren

(Realität)

$$f(x) := x + 5$$

```
PROGRAM f(INPUT, OUTPUT);

FUNCTION f(x : INTEGER) : INTEGER;
BEGIN
   f := x + 5;
END;

VAR x :  INTEGER;

BEGIN
    ReadLn(x) ;
    WriteLn(f(x))
END.
```

Programmieren

(Realität)

$$f(x) := x + 5$$

```
PROGRAM f(INPUT, OUTPUT);

FUNCTION f(x : INTEGER) : INTEGER;
BEGIN
   f := x + 5;
END;

VAR x :  INTEGER;

BEGIN
    ReadLn(x) ;
    WriteLn(f(x))
END.
```

Programmieren

(Realität)

- orientiert an Maschine
- 10% Problemlösung, 90% Syntax/Maschine

Informatik-Curriculum Baden-Württemberg

- Syntax
- Ein-/Ausgabe
- Zahlen, Zeichenketten
- Arithmetik
- Variablen und Zuweisung
- Anweisungsfolge, Verzweigung, Wiederholung
- Prozeduren und Funktionen
- Rekursion in einfachen Fällen
- strukturierte Datentypen

Programmieren und Mathematik

„Ist n Element der Liste l?“

$$g(n,l) := \begin{cases} false & \text{falls } l \text{ leer} \\ true & \text{falls } n = \mathrm{first}(l) \\ g(n, \mathrm{rest}(l)) & sonst \end{cases}$$

Realität

```
TYPE
   ListType = ^NodeType;
   NodeType = RECORD
                 First : String;
                 Rest  : ListType
              END;

FUNCTION IsMember (Name : String;  List : ListType)
: BOOLEAN;
BEGIN
   IF List = NIL
      THEN IsMember := FALSE
   ELSE IF Name = List^.First
      THEN IsMember := TRUE
   ELSE
      IsMember := IsMember(Name, List^.Rest)
END;
```

Realität

```
PROGRAM NameOnList(Input, Output) ;
VAR List : ListType;
    Name : STRING;

PROCEDURE GetList(Var LIST : ListType);
BEGIN
  ...
END;

FUNCTION IsMember ...

BEGIN
    ReadLn(Name);
    GetList(List);
    WriteLn(IsMember(Name, List))
END.
```

Allein am Computer

```
...
y + 1 := x;
...
```

```
warning: expression used as a statement - value is ignored
parse error before ‘+’
```

Realität

- von-Neumann-Modell statt Algebra
- Syntax statt Prinzipien
- Zeitverschwendung mit Ein-/Ausgabe
- Basteln statt systematisches Problemlösen
- Werkzeuge, die für professionelle Programmierer gedacht sind

Leidiges Thema Rekursion

Fakultät, Potenz, Fibonacci . . .

Bausteine

- Programmiersprache Scheme
- Programmierumgebung speziell für Anfänger (kostenlos zum Herunterladen)
- systematische Konstruktionsanleitungen

Inhalte

- Algebra, Funktionen
- Fallunterscheidungen
- Konstruktionsanleitungen
- Symbole
- Listen
- Strukturen („Records“)
- Grafik
- selbstbezügliche Strukturen

Datengesteuerte Programmkonstruktion

Daten $\Longrightarrow$ Programmschablone

selbstbezügliche Daten $\Longrightarrow$ rekursive Programmschablonen

„Eine Liste von Symbolen ist ...

- die leere Liste `empty`, oder
- `(cons` s l`)`, wobei s ein Symbol und l eine Liste von Symbolen ist.“

Rekursion Nebenprodukt des Datenmodells

Ziele

- Programmieren als
  präzise Kommunikation über Problemlösungen
- Programmieren als Problemlösen
- Programmieren als Training für Mathematisierung

Voraussetzungen

- weniger Ketchup
- Werkzeuge für Anfänger, nicht professionelle Programmierer
- systematische Didaktik
  mit nachvollziehbaren Anleitungen
- ansprechende Beispiele

Zusammenfassung

- Programmierausbildung birgt große Chancen
- … aber:
  - mehr Ketchup als Kaviar
  - anfängeruntaugliche Software
  - Didaktik orientiert sich an Maschine, nicht am Anfänger