Programmiersprachen im Informatikunterricht

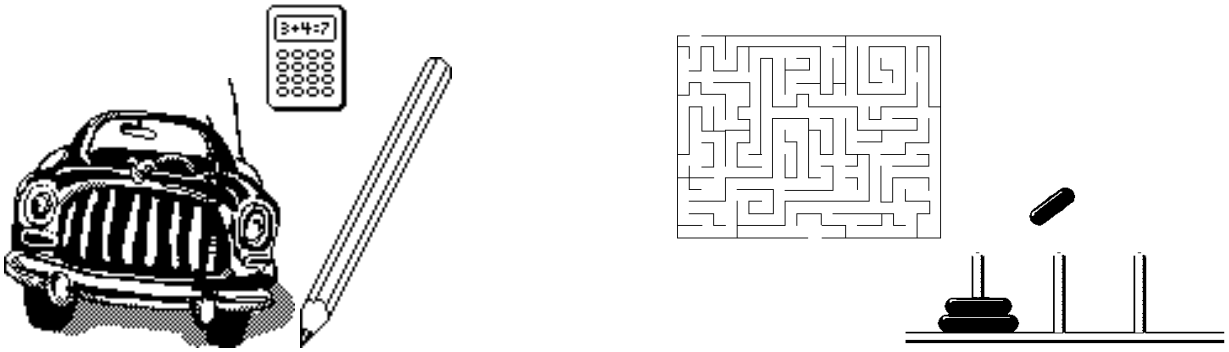
Andreas Schwill

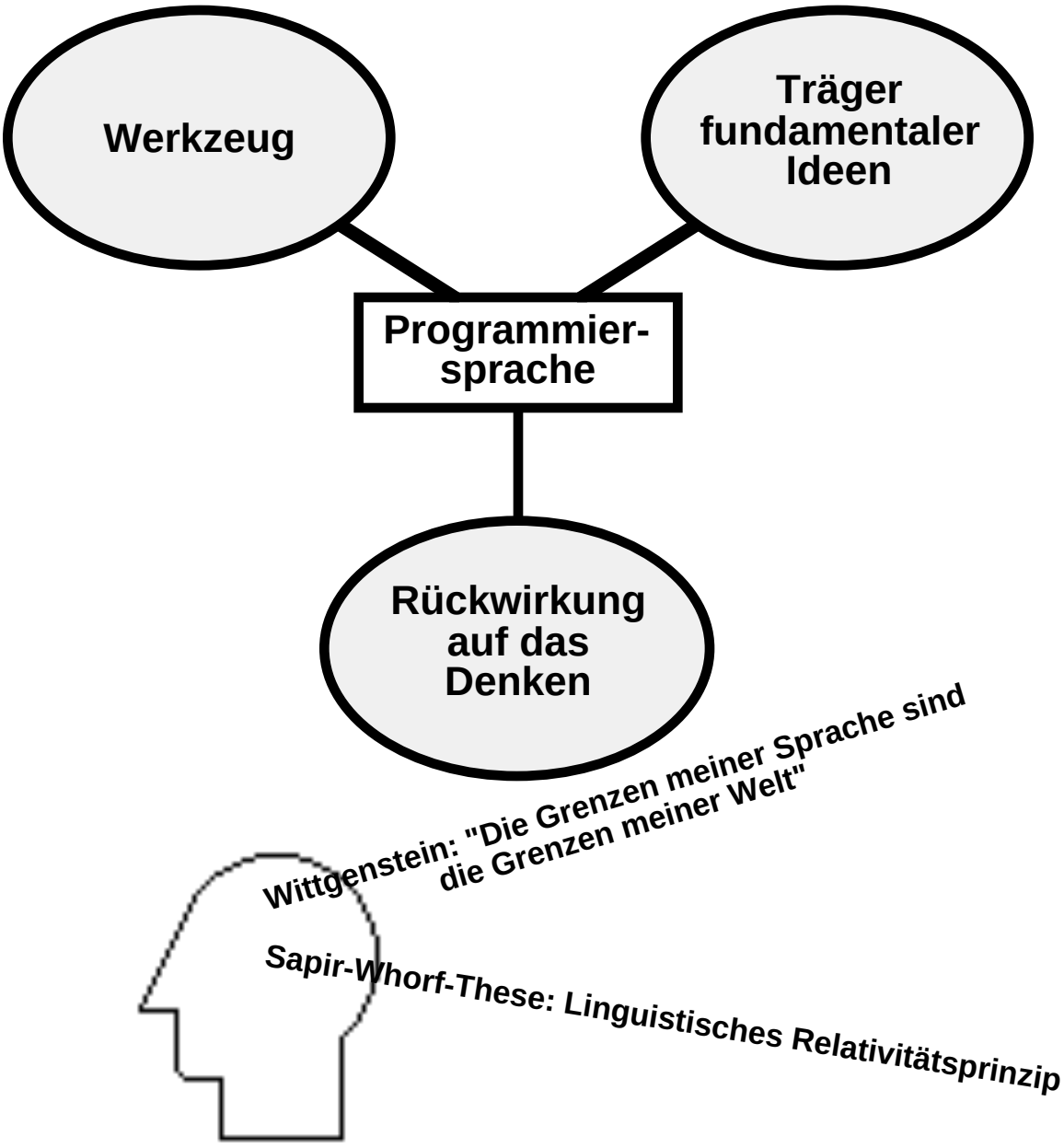
Institut für Informatik

Überblick

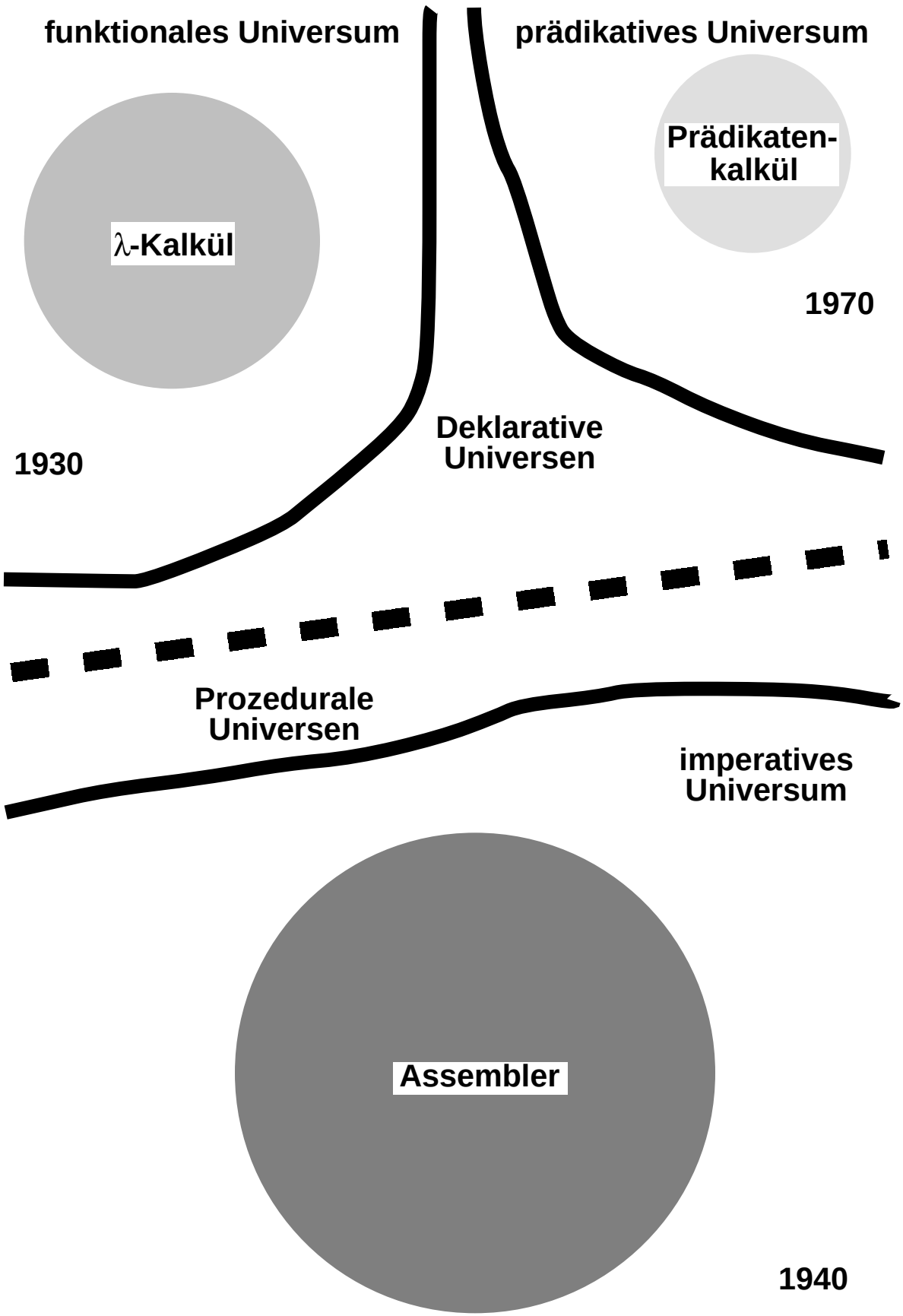
- **Bedeutung von Programmiersprachen**
- **Informatische Komponente**
  - **Die Welt der Programmiersprachen**
  - **Merkmale der Programmiersprachen**
- **Didaktische Komponente**
  - **Der Sprachenstreit im Informatikunterricht**
  - **Programmiersprachen im unterrichtlichen Vergleich**
  - **Kognitive Aspekte objektorientierter Programmierung**
  - **Methodische Hinweise**

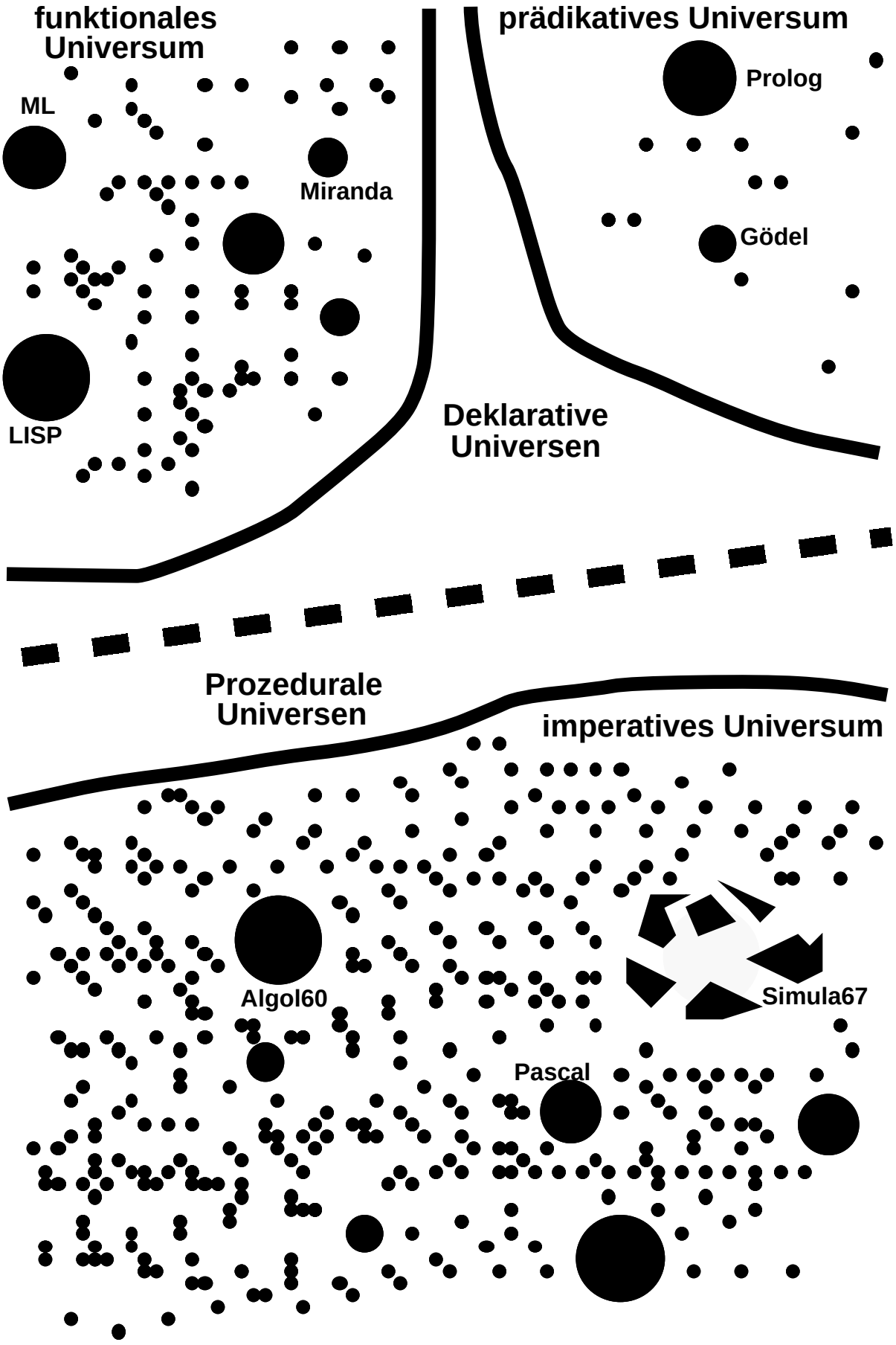
1 Bedeutung von Programmiersprachen





2 Die Welt der Programmiersprachen




funktionales Universum
ML
Miranda
LISP
prädikatives Universum
Prolog
Gödel
Deklarative Universen
Prozedurale Universen
imperatives Universum
Algol60
Simula67
Pascal

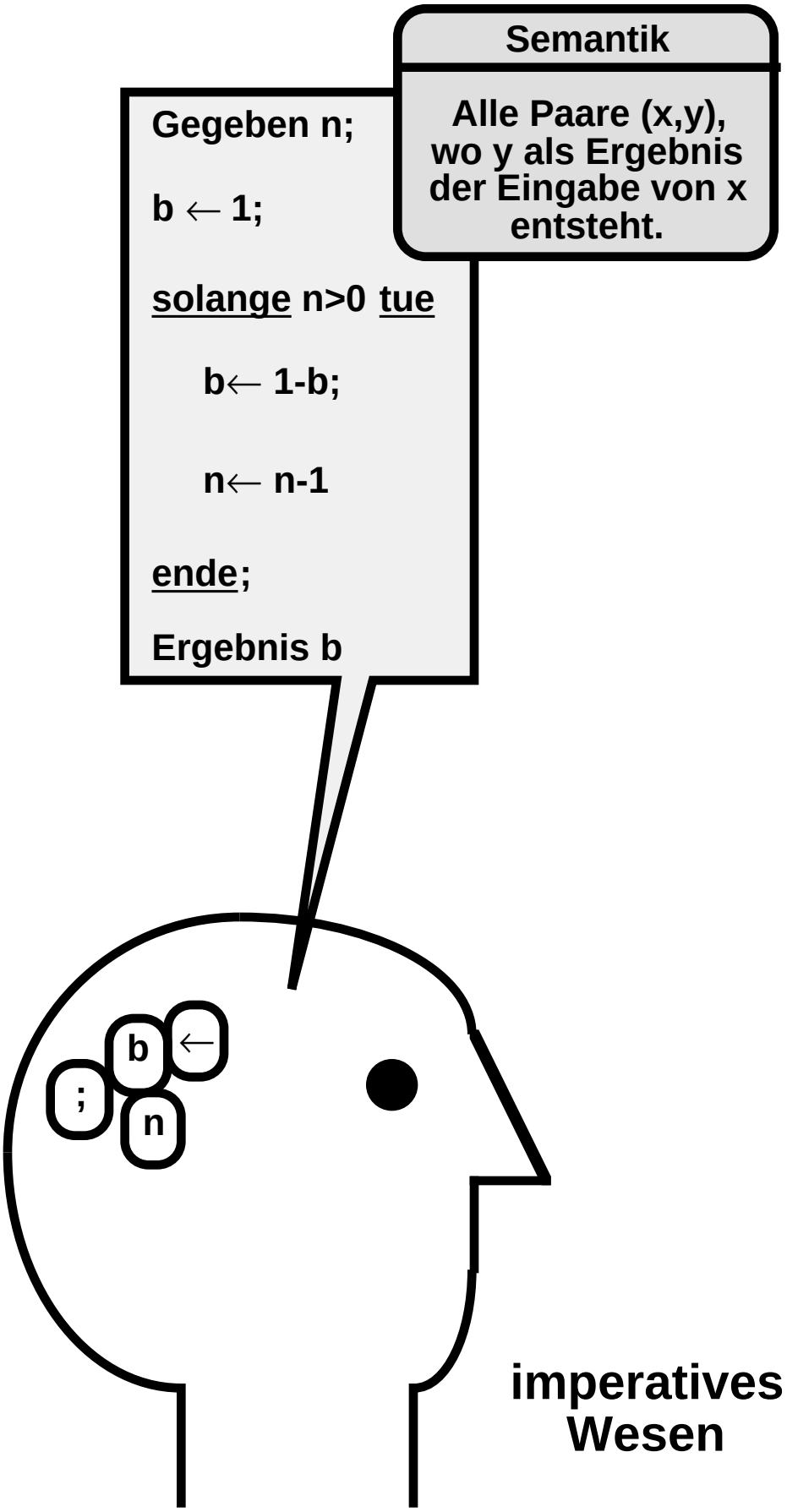
3 Merkmale der Programmiersprachen

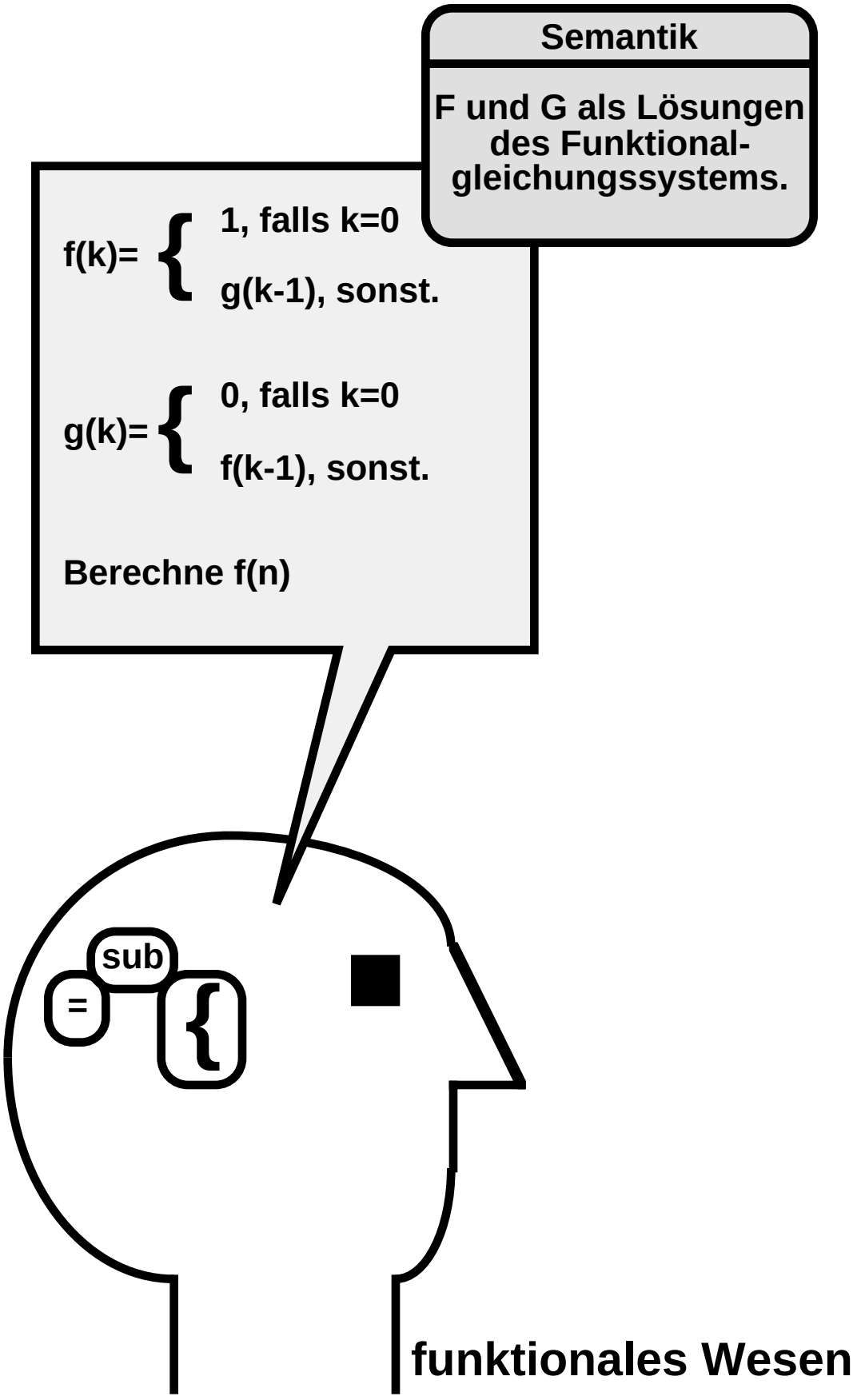
Wissen	Beispiele
Exakte Beschreibung allgemeiner Eigenschaften von Objekten sowie ihrer Beziehungen untereinander. **(Zweckfreiheit)**	• **Katzen trinken Milch** • **Das Quadrat einer geraden Zahl ist gerade.**

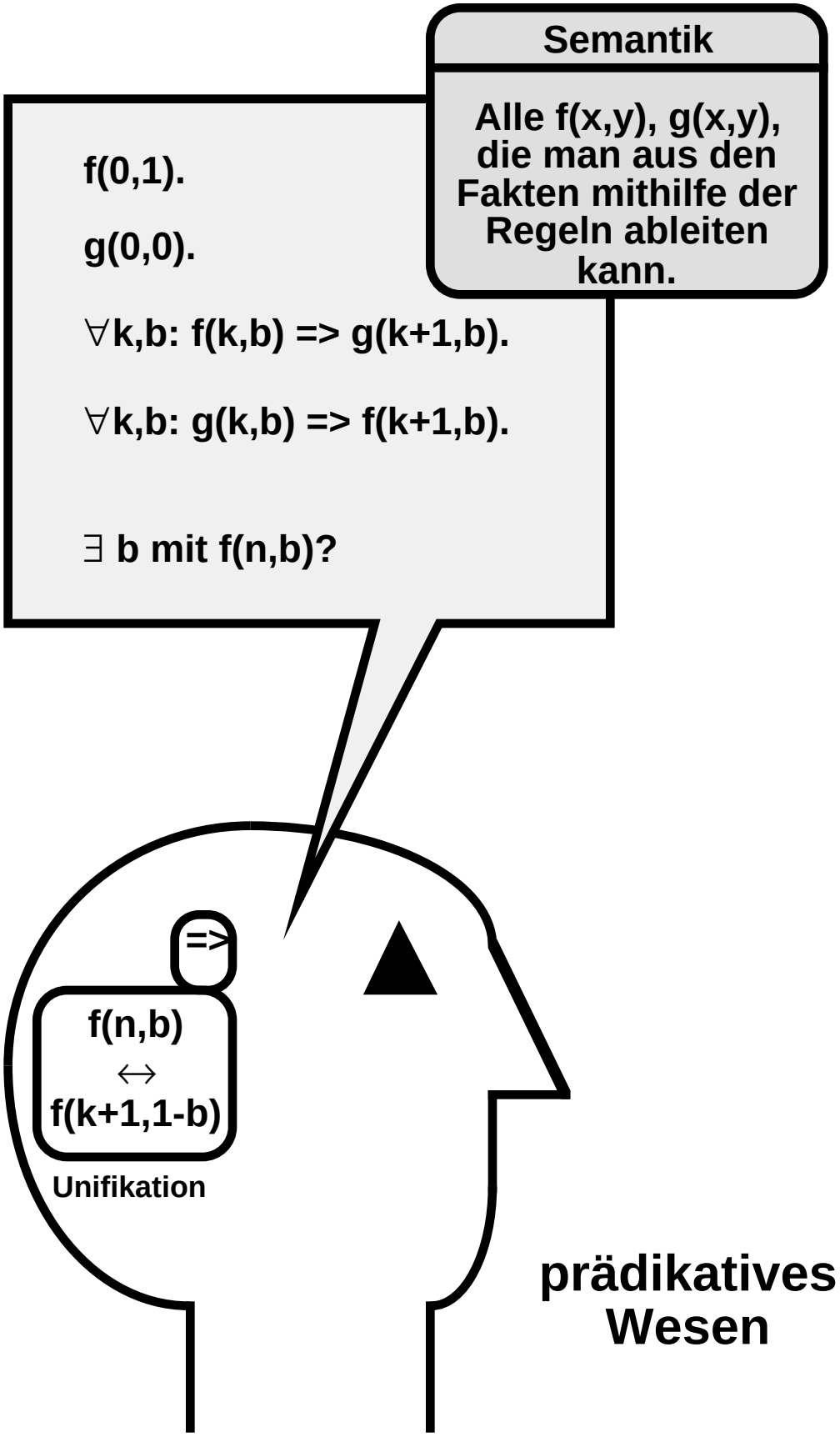
Deklarative Universen

Prozedurale Universen

Lösungswege	Beispiele
Exakte Festlegung, auf welchem Weg die Lösung zu einem Problem maschinell zu berechnen ist. **-> gegebene Größen** **-> gesuchte Größen** **-> Lösungsweg** **(Zweckhaltigkeit)**	• **Zum Öffnen Lasche anheben, zusammendrücken und farbige Ecke abreißen.** • **Falls Sie weitere Informationen wünschen, senden Sie bitte den ausgefüllten Coupon zurück.**

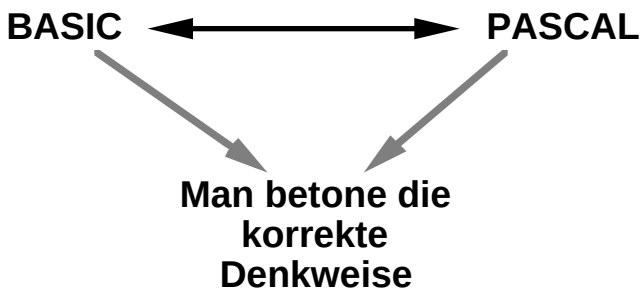

Semantik
Alle Paare (x,y), wo y als Ergebnis der Eingabe von x entsteht.
Gegeben n;
b ← 1;
solange n>0 tue
b← 1-b;
n← n-1
ende;
Ergebnis b
b
←
;
n
imperatives Wesen


Semantik
F und G als Lösungen des Funktional-gleichungssystems.
f(k)= { 1, falls k=0
g(k-1), sonst.
g(k)= { 0, falls k=0
f(k-1), sonst.
Berechne f(n)
sub
=
{
funktionales Wesen

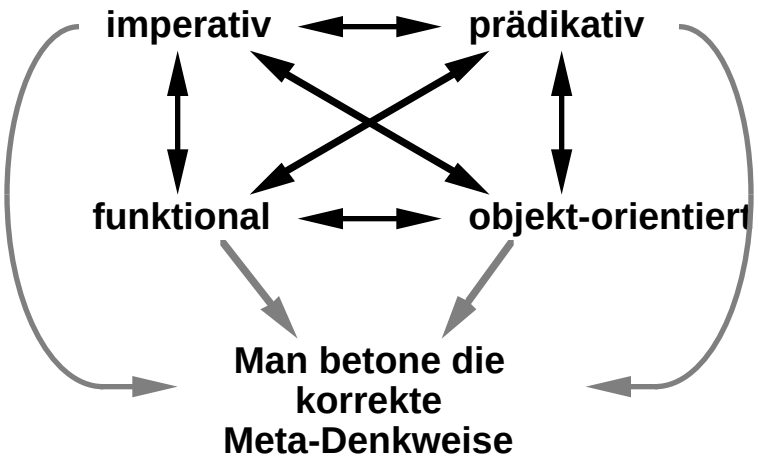

Semantik
Alle f(x,y), g(x,y), die man aus den Fakten mithilfe der Regeln ableiten kann.
f(0,1).
g(0,0).
∀k,b: f(k,b) => g(k+1,b).
∀k,b: g(k,b) => f(k+1,b).
∃ b mit f(n,b)?
=>
f(n,b)
↔
f(k+1,1-b)
Unifikation
prädikatives Wesen

4 Der Sprachenstreit im Informatikunterricht

1975: Welche Programmiersprache ist für die Schule am geeignetsten?



1990: Welche Denkweise ist für die Schule am geeignetsten?



2005: Welche Meta-Denkweise (??) ist die geeignetste?

...

5 Programmiersprachen im unterrichtl. Vergleich

Imperative Programmierung mit Pascal

Erfahrungen:

+ dominiert den Informatikunterricht
+ gute Lernerfolge auf lange Sicht
+ besitzt Bezug zur Lebenswelt der Schüler
+ ist in der Informatik im Zusammenhang mit Rechnermodellen und Effizienzanalysen unverzichtbar

Mängel:

- umfangreiche Syntax
- (zu) viele (zu) elementare Konzepte
- Mangel an Orthogonalität (Baukastenprinzip)
- Motivationsprobleme zu Beginn
- Programme mit Wegwerf-Charakter
- Lernen "auf Vorrat"

Prädikative Programmierung mit PROLOG

Erfahrungen:

+ viele interessante Vorschläge, PROLOG in den Anfangsunterricht einzubeziehen
+ kein Motivationsproblem: Schüler können sehr früh mächtige Programme schreiben
+ hohes Maß an Orthogonalität

Mängel: Alle Vorschläge setzen voraus,

- daß PROLOG eine große Nähe zur natürlichen Sprache besitzt,
- daß Anfänger leichter Probleme beschreiben als Vorschriften zu ihrer Lösung angeben können, im Widerspruch zu empirischen Ergebnissen.

"**Nähe zur natürlichen Sprache**": Alltagslogik=Prädikatenlogik.

1. **Problem**: Umsetzung umgangssprachlicher Aussagen mit temporalen, kausalen oder undifferenzierten logischen Operationen in prädikatenlogische Aussagen:

   Sie war reich, und er heiratete sie <=> Er heiratete sie, und sie war reich.

   Paula ist eine gute Schülerin, weil sie hart arbeitet.

   Wenn es regnet, nehme ich einen Schirm

   => Wenn es nicht regnet, nehme ich keinen Schirm.

2. **Problem**: closed world assumption versus open world reality:

   Mensch: weiß nicht

   Gibt es auf Tahiti Vulkane?

   PROLOG: nein

3. **Problem**: Deklarative <=> prozedurale Semantik.
   Anfänger benötigen Verständnis für virtuelle PROLOG-Maschine.

"**Problemlösen durch Problembeschreiben**":

Idee: zweckfreies Wissen

Computer Lösung

Problem

Tatsächlich: Eingegebenes Wissen ist in hohem Maße zweckorientiert.

Konsequenzen:

- Schüler sind nicht gut in Logik

  => Einführung in Prädikatenlogik nötig
- Schüler können Modellierungprobleme nicht intuitiv lösen

  => Vertieftes Verständnis von PROLOG und der virtuellen PROLOG-Maschine nötig
- Vorteile von PROLOG im Anfangsunterricht sehr zweifelhaft

Funktionale Programmierung

Erfahrungen:

- kaum Vorschläge und Ergebnisse mit modernen Sprachen
- positive Erfahrungen mit LOGO evtl. nicht übertragbar
- mathematische Notation für Anfänger evtl. zu abschreckend
- Psychologie: Rekursion schwierig für Anfänger

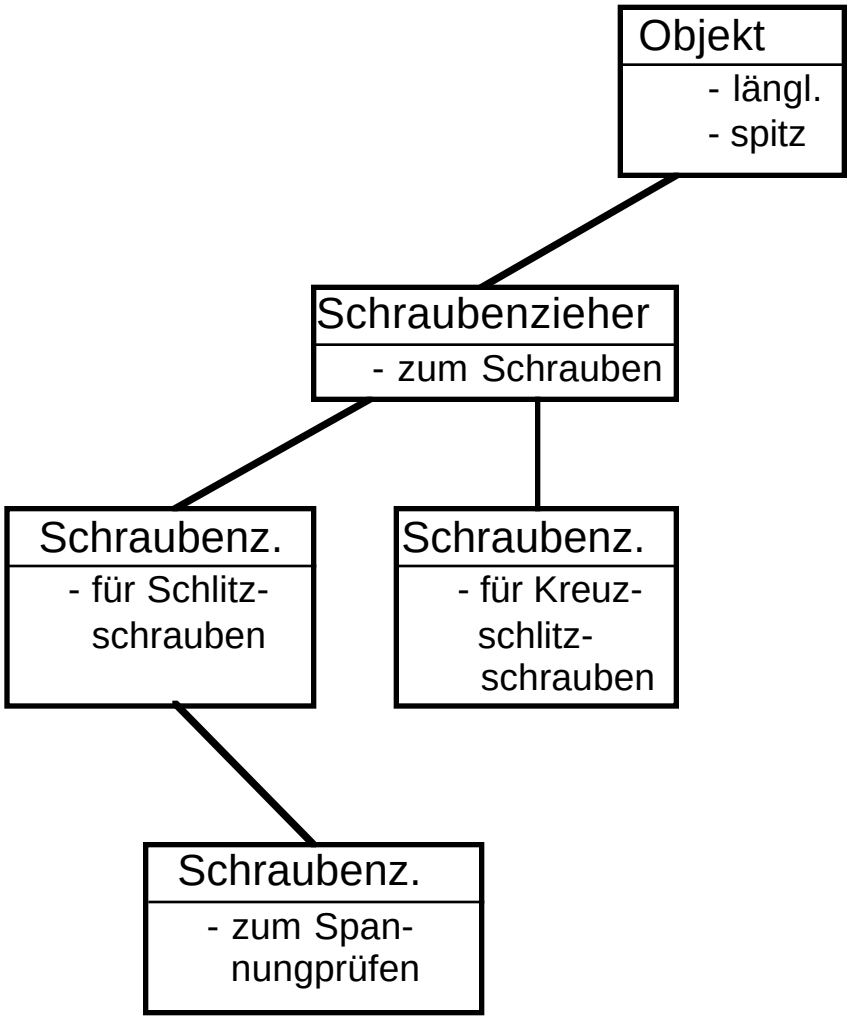
Objektorientierte Programmierung

Erfahrungen:

+ erfüllt Forderungen nach einem zeitgemäßen Unterricht mit modernen Konzepten, wie Erweiterbarkeit, Vererbbarkeit, Kapselung etc.
+ erfüllt Wünsche nach einer stärkeren Anwendungsorientierung durch Nutzung des Computers anstelle von seinem vertieften Verständnis
+ erfüllt das Prinzip der Fortsetzbarkeit eines nach dem Spiralprinzip organisierten Unterrichts;
  Umstellung von speziellen Anfängersystemen wie LOGO oder Roboter NIKI auf eine "echte" Sprache entfällt
+ paßt zu den psychologischen Prozessen, die im menschlichen Gehirn beim Denken, Erkennen und Problemlösen ablaufen,
+ paßt damit insbesondere zu den kognitiven Voraussetzungen von Anfängern.

6 Kognitive Aspekte objektorientierter Programmierung

Objekte aus kognitiver und informatischer Sicht



Ergebnisse der Kognitionspsychologie:

Menschen identifizieren Objekte
mehr durch die Aktionen, die mit ihnen möglich sind,
als durch ihr Aussehen wie Farbe oder Form.

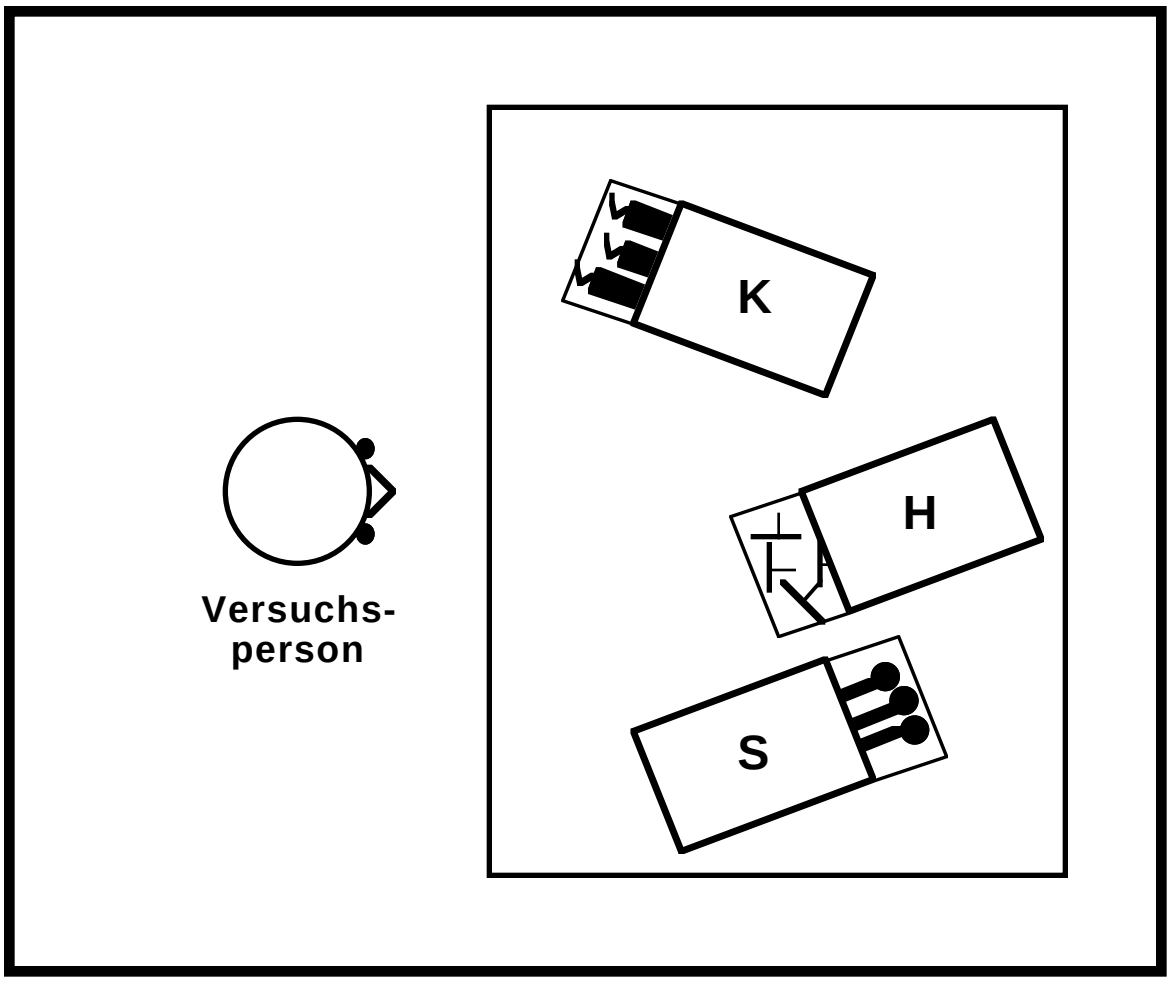
Standardmodell der Kognitionspsychologie: Kategorie/Schema

Analogie: Klasse <-> Kategorie/Schema

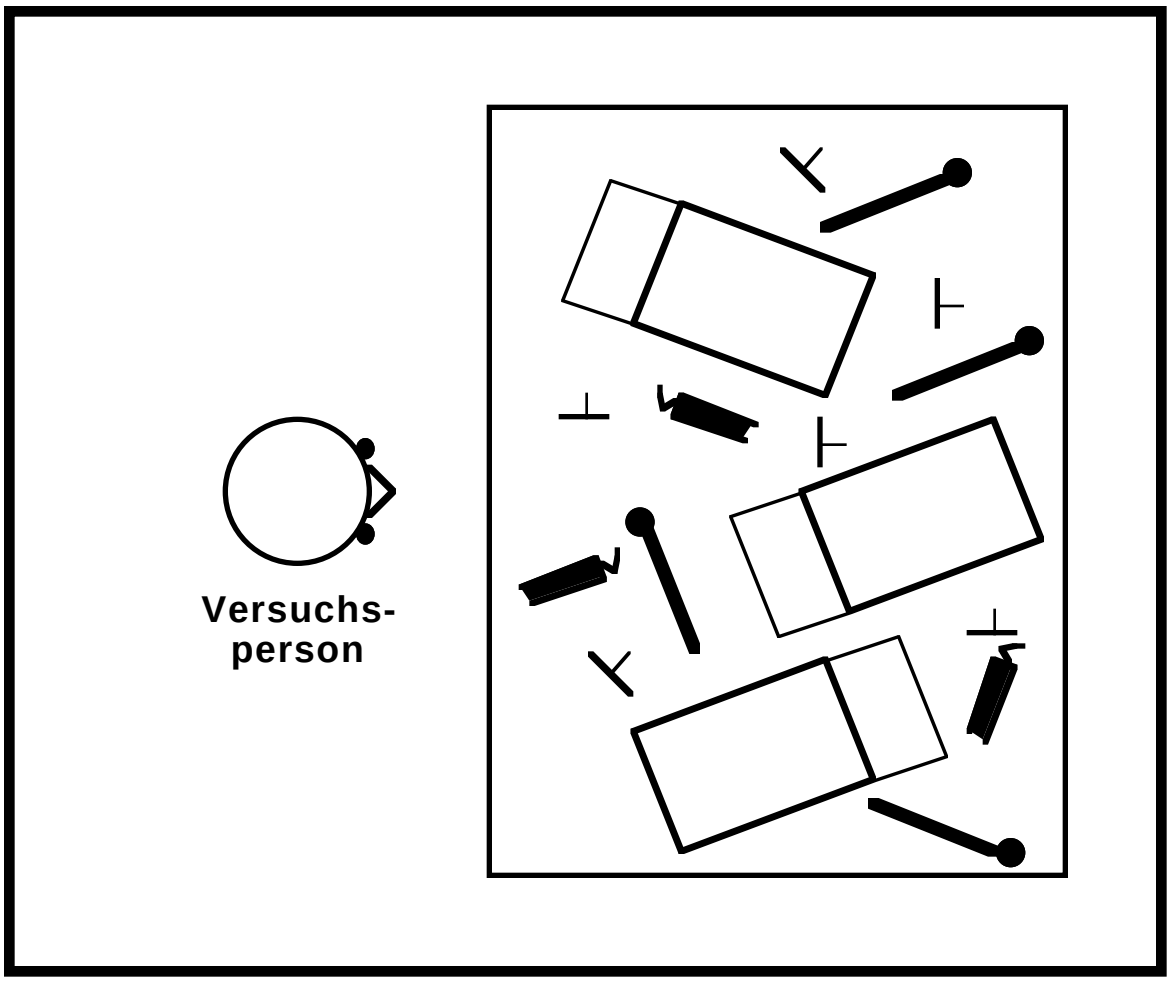
objektorientierte Sicht	Beispiel	psychologische Sicht
Klasse	Hund	Kategorie/Schema = große komplexe Einheit, die große Teile menschl. Wissens und Verhaltens organisiert
Definition durch **Attribute**:		Definition durch **Attribute**:
• Variablen	hat 4 Beine	• Wahrnehmungsattribute
• Methoden	kann bellen	• Funktionsattribute
• Vererbung		• relationale Attribute
•• einfach	Hund ist Haustier	•• Kategorien können sich gegenseitig ausschließen
•• mehrfach	Geparden verbinden Eigenschaften von Hund und Katze	•• Kategorien können sich überlappen

Beispiel: Das Kerzenproblem von K. Duncker

Situation 1



Situation 2



Beobachtung:

- **Versuchspersonen 1 nehmen die Schachteln als Behälter wahr; Die Funktion "Behälter" beschränkt die nachfolgenden Denkabläufe (funktionale Gebundenheit)**

- **Versuchspersonen 2 nehmen die Schachteln ohne Bindung an irgendeine Funkton wahr.**

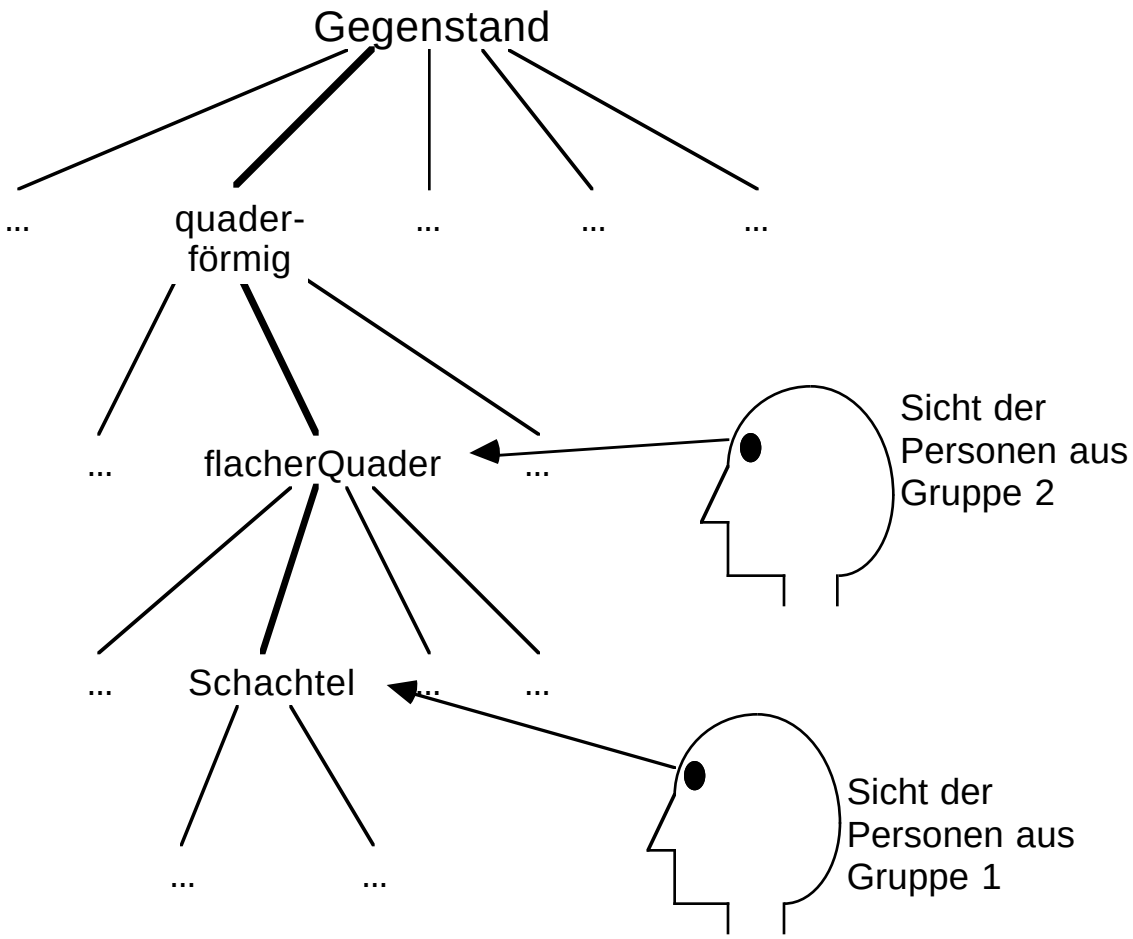
Eine objektorientierte Interpretation (OBERON-artig)

type **Gegenstand** = record
    hat eine räuml. Ausdehnung;
    kann man greifen;
    hat Farbe und Form;
    …
end;

type **quaderförmig** = record (**Gegenstand**)
    hat Länge, Breite, Höhe;
    …
end;

type **flacherQuader** = record (**quaderförmig**)
    Höhe≤5cm;
    kann man stapeln;
    kann man als Unterlage verwenden;
    …
end;

type **Schachtel** = record (**flacherQuader**)
    kann man öffnen;
    kann man schließen;
    ist leer oder gefüllt mit ...;
    …
end.


Gegenstand
...
quader-
förmig
...
...
...
...
flacherQuader
...
Sicht der
Personen aus
Gruppe 2
...
Schachtel
...
...
...
...
Sicht der
Personen aus
Gruppe 1

7 Methodische Hinweise

Objektorientierte Programmierung

- **ist aus Sicht der Informatik ein moderner und leistungsfähiger Ansatz,**
- **spiegelt fundamentale kognitive Prozesse im menschlichen Gehirn wider,**
- **ist folglich besonders geeignet für den einführenden Informatikunterricht,**

vorausgesetzt, daß

- **man eine Programmierumgebung wählt, die die objekt-orientierte Vorgehensweise sichtbar macht, d.h.**
  = eine künstliche Welt generiert,
  = viele, klar visualisierte Standardobjekte bietet,
  = eine Benutzungsschnittstelle besitzt, mit der man spielerisch explorativ Objekte manipulieren, analysieren, kombinieren, rekonfigurieren, evolutionär erweitern, neu entwickeln kann.

Systeme, die dieser Philosophie nahekommen, sind z.B.:

HyperCard	**für Anfängerniveau (mit fester Klassenhierarchie, vereinfachtem Nachrichtenaustausch, HyperTalk)**
Smalltalk-80	**für Fortgeschrittene.**