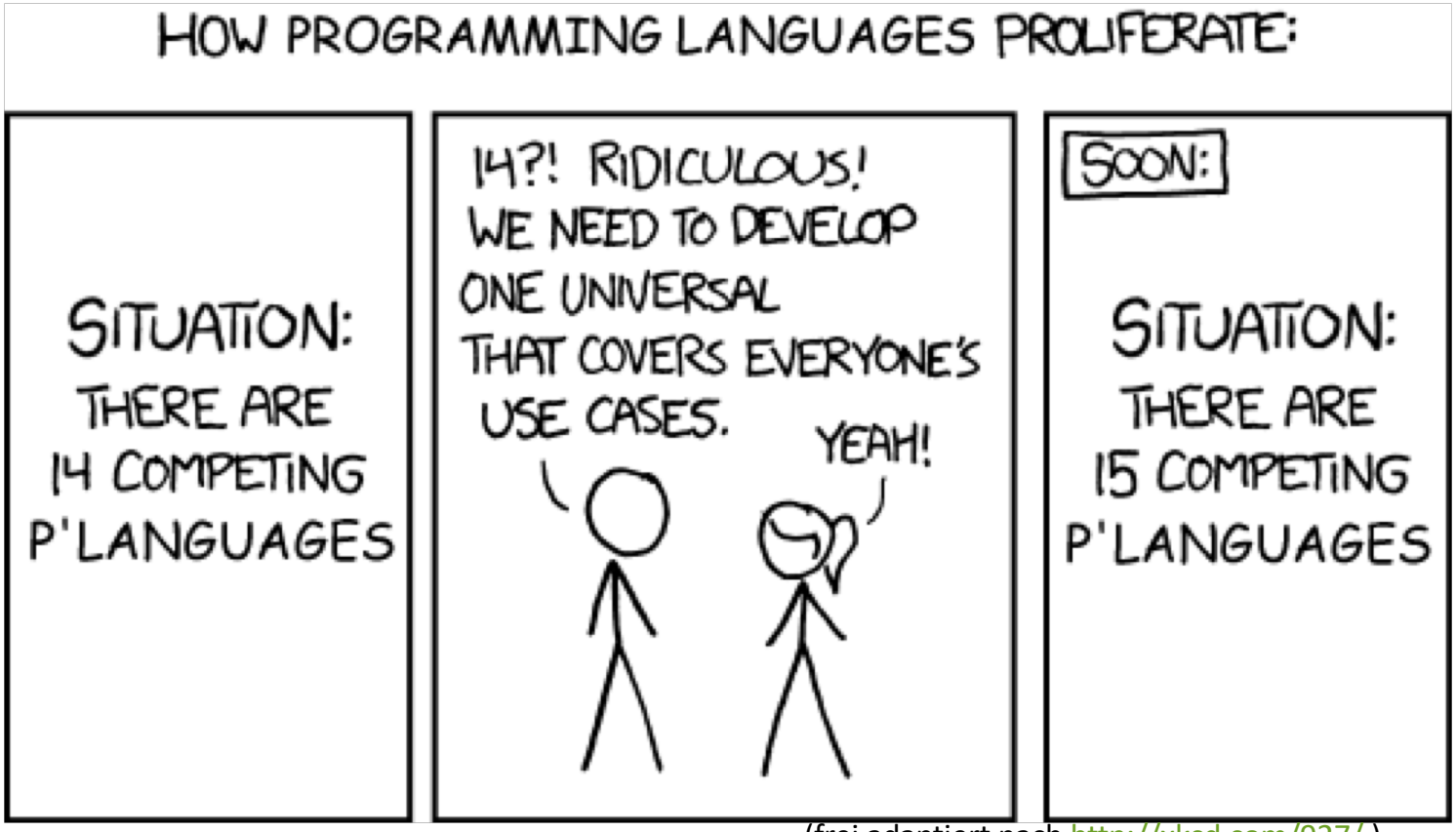
K01

Einführung – Teil I



(frei adaptiert nach http://xkcd.com/927/)

Verbreitete Programmiersprachen

- Etwa 2500 Programmiersprachen bekannt[1]
- TIOBE programming community index[2]
  - Indikator für Popularität einer Programmiersprache
  - Misst nicht
    - welche die *beste* ist,
    - oder welche die *meisten* lines of code (LOC) hat.
- Andere Indices
  - Language Popularity Index[3]
  - PYPL – PopularitY of Programming Language[4]

[1] http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm

[2] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[3] http://lang-index.sourceforge.net/

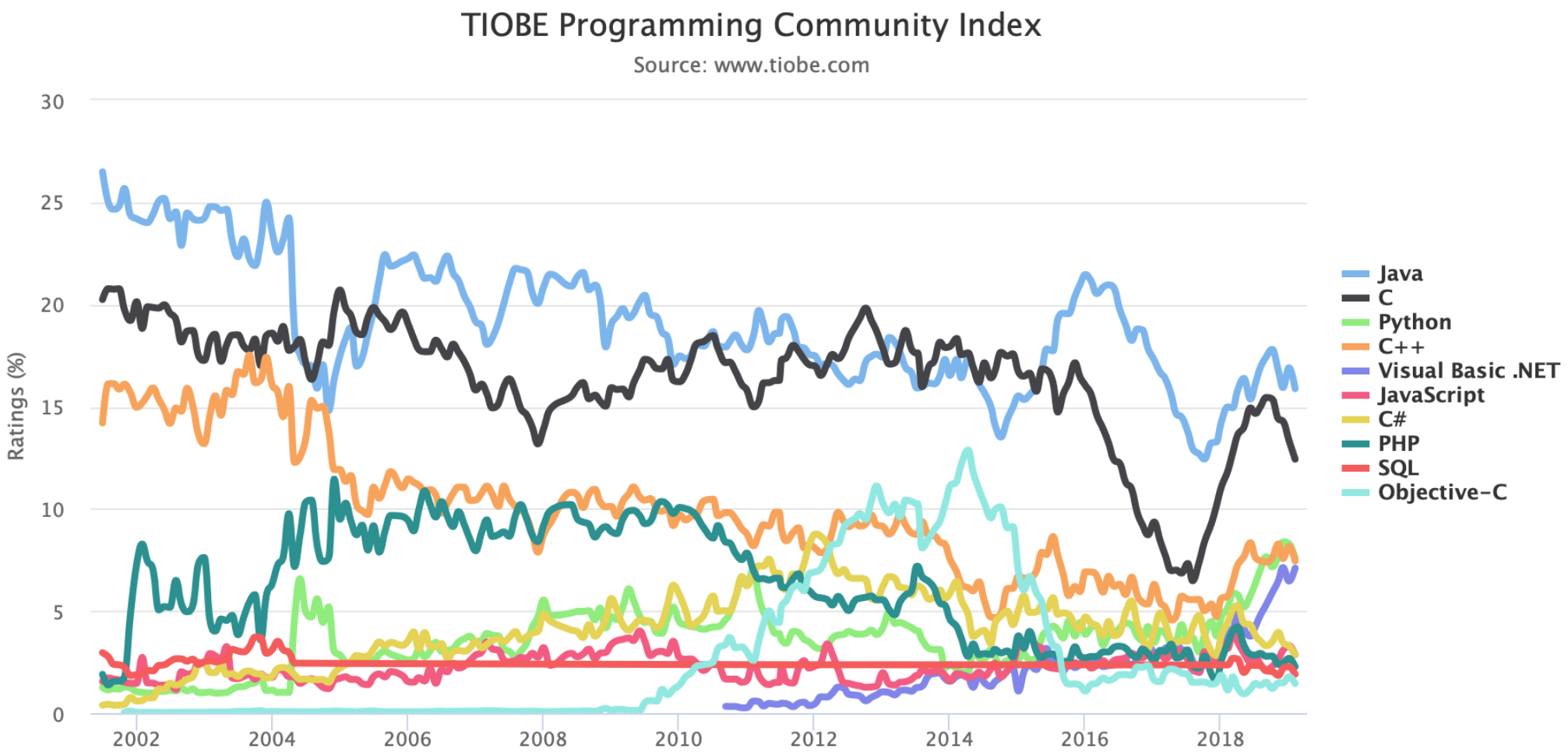
[4] https://sites.google.com/site/pydatalog/pypl/PyPL-PopularitY-of-Programming-Language

TIOBE Top 20 Stand: 02/2019

Feb 2019	Feb 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.876%	+0.89%
2	2		C	12.424%	+0.57%
3	4	↑	Python	7.574%	+2.41%
4	3	↓	C++	7.444%	+1.72%
5	6	↑	Visual Basic .NET	7.095%	+3.02%
6	8	↑	JavaScript	2.848%	-0.32%
7	5	↓	C#	2.846%	-1.61%
8	7	↓	PHP	2.271%	-1.15%
9	11	↑	SQL	1.900%	-0.46%
10	20	⇈	Objective-C	1.447%	+0.32%
11	15	⇈	Assembly language	1.377%	-0.46%
12	19	⇈	MATLAB	1.196%	-0.03%
13	17	⇈	Perl	1.102%	-0.66%
14	9	⇊	Delphi/Object Pascal	1.066%	-1.52%
15	13	↓	R	1.043%	-1.04%
16	10	⇊	Ruby	1.037%	-1.50%
17	12	⇊	Visual Basic	0.991%	-1.19%
18	18		Go	0.960%	-0.46%
19	49	⇈	Groovy	0.936%	+0.75%
20	16	⇊	Swift	0.918%	-0.88%

Scala?
TypeScript?
Haskell: 39 (43)
Prolog: 35 (37)

TIOBE - Langzeitverlauf



Github-Programmiersprachenstatistik

http://githut.info

Was ist ein Programmierparadigma?

Es existiert keine präzise, formale (mathematische) Definition. **ill-defined**

Duden Fremdwörterbuch: griechisch *parádeigma - Beispiel, Muster*

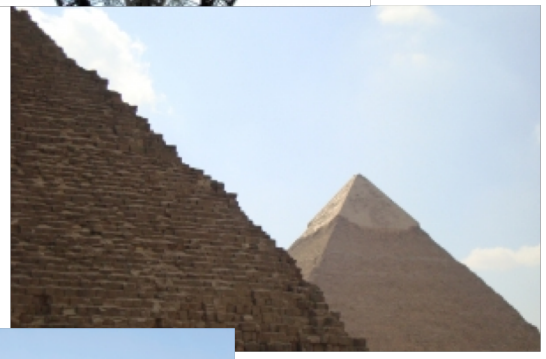
Wikipedia: *Set of practices that define a scientific discipline during a particular period of time.*

Heute wird der Begriff im Sinne von

Denkmuster, bzw. das *Prinzip nach dem man vorgeht*

benutzt.

Begriff des *Programmierparadigma* und Begriff des *wissenschaftlichen Paradigma* sind verwandt. Letzteres wird nach T. S. KUHN als eine *Lehrmeinung* aufgefasst. Neue Erkenntnisse führen i.d.R. zur Aufgabe veralteter Lehrmeinungen (Umbruch = Paradigmenwechsel). Bei Programmierparadigmen kann dies prinzipiell auch passieren (wenn auch noch nicht geschehen).



Analogie zur Architektur

- Holzbau: man denkt in Balken und Brettern
- Ziegelbau: man kalkuliert mit der Anzahl, Länge, Stärke von Ziegeln
- Manche Gebäude sind nur nach einem Paradigma errichtbar: Hochhäuser können z.B. nur bis zu einer bestimmter Grösse aus Holz errichtet werden
- Wichtig: Architekt soll die verschiedenen Paradigmen kennen und gezielt einsetzen, je nach Zweckmässigkeit für das jeweilige Ziel.

Progr'paradigma – 2er Versuch

*„Ein Programmierparadigma ist das einer Programmiersprache oder einer Programmiertechnik zu Grunde liegende Prinzip entweder zur Beschreibung von Arbeitsabläufen für einen Computer oder zur Beschreibung der Aufgabenstellung.“**

* Quelle: P. Forbig, I.O. Kerner. Programmierung – Paradigmen und Konzepte.

Progr'paradigma – 3er Versuch

Sei $\mathbf{C}$ *die endliche Menge aller bekannten Programmiersprachen-konzepte; z.B. call-by-value, call-by-need, statisch typisiert, Operator-überladung, referentielle Transparenz, strikte Auswertung, Pointer, Templates, Multiple Dispatch, strukturelle Typisierung ...*

Syntax, kognitive Aspekte, Vorlieben und prinzipbedingte Eigen-schaften ausser Acht lassend, ist ein Programmierparadigma $\mathbf{P}$ *eine nichtleere Teilmenge* $\mathbf{P} \subseteq \mathbf{C}$.

Zwei Programmierparadigmen $\mathbf{P_1}$, $\mathbf{P_2}$ können disjunkt sein ($\mathbf{P_1} \cap \mathbf{P_2} = \emptyset$), oder in Teilen dieselben Konzepte beinhalten ($\mathbf{P_1} \cap \mathbf{P_2} \neq \emptyset$).

Klassifikation von Progr'sprachen

Ziel (jeder Klassifikation):

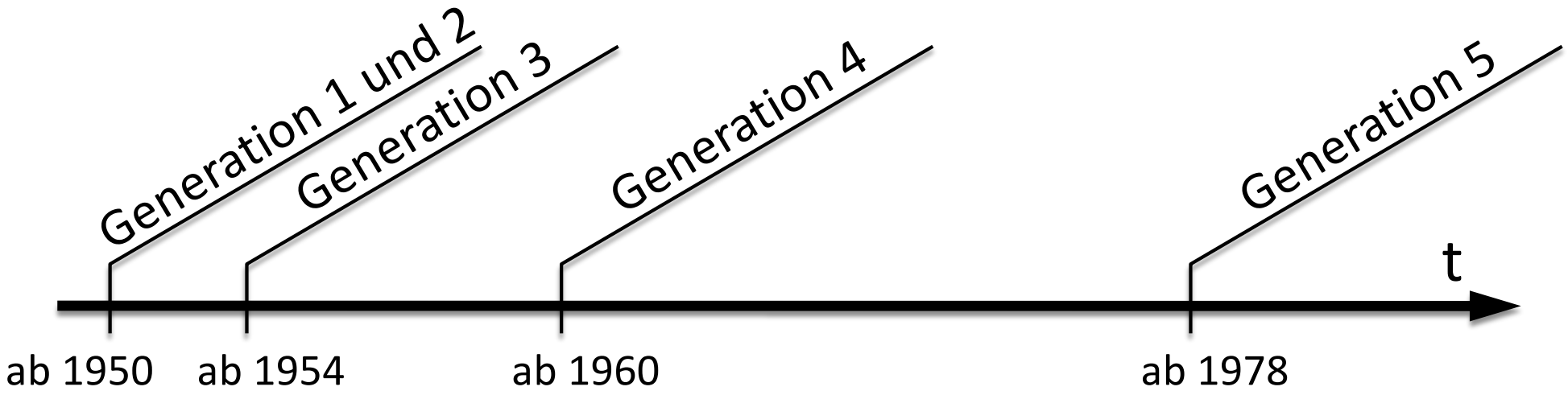
- Verstehen der Gemeinsamkeiten & Unterschiede
- Liefert:
  - Abgrenzung, Ordnung
  - Vokabular bzw. Terminologie – in unserem Fall sind dies Programmiersprachen...

...generationen (zeitlich)

...konzepte (ideell)

...klassen (kategorisch)

Klassifikation nach Generation



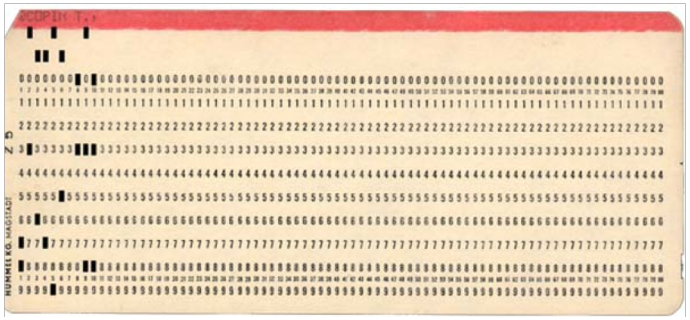
Grenzen unscharf

1957 FORTRAN, 1958 ALGOL, 1960 LISP, 1960 COBOL, 1967 SIMULA
1964 BASIC, 1970 Pascal, 1972 C, 1972 Prolog, 1975 Scheme,
1980 Smalltalk-80, 1980 Ada, 1983 C++, 1986 Eiffel, 1988 Mathematica,
1990 Haskell, Java 1995

Generation 1 und 2

Generation 1:

- Binäre, prozessorspezifische Codes
- Meist kombiniert mit zu verarbeitenden Daten



```
1011 0000  0110 0011
```

(hex `B0 63`; lade den Wert `63h` in Register `AL`, Intel x86 CPU)

Generation 2:

- Assemblersprachen
- Codes werden ersetzt durch benannte Befehle (Mnemonics)
- Benötigen schon (einfachen) Compiler

```
mov al, 63h
```

(Intel Syntax)

Generation 3 bis 5

Generation 3:

- „höhere“ Programmiersprachen
- Abstraktion vom Prozessorbefehlssatz

Generation 4:

- Keine klar erkennbare Grenze zur 3. Generation
- Wurde oft als Marketinginstrument verwendet
- Zahlreiche nicht-Turing-vollständiger Sprachen (z.B. SQL)
- Zunehmende Bedeutung von Werkzeugen (IDE, GUI builder)

Generation 5:

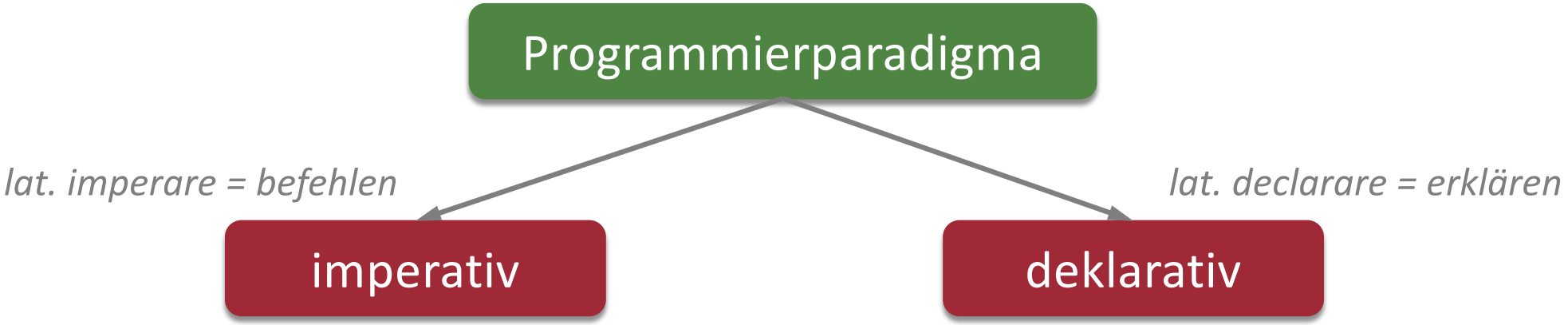
- Probleme werden durch Rand- bzw. Zwangsbedingungen implizit beschrieben
- System muss selbst eine Lösung finden/suchen unter Einhaltung der Bedingungen; siehe auch nachfolgende Klassifikation

Stammbaum

Einen chronologischen Überblick der wichtigsten Programmiersprachen liefert:

http://www.levenez.com/lang/lang.pdf

Klassifikation nach Prinzip



Programmierer beschreibt Ablauf

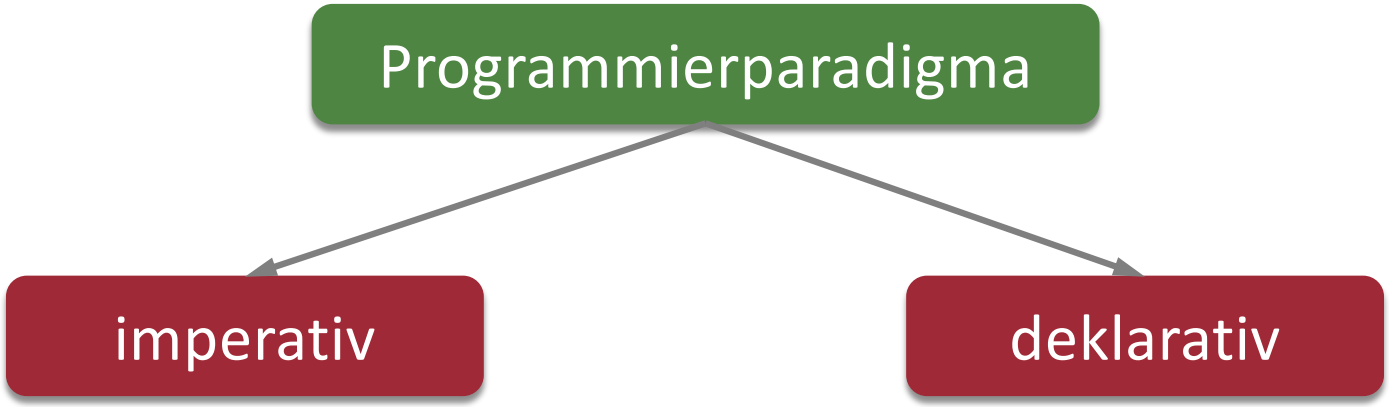
- **wie** ist das Problem zu lösen
- Schritt für Schritt
- mittels **Anweisungen** der Programmiersprache; nicht notwendigerweise der Maschine

Programmierer beschreibt Problem

- **was** ist das Problem
- Basis math. Kalkül bzw. Theorie
- Problem + Kalkül ergibt ausführbare Lösungsschritte
- abstrahiert immer von Maschine

Viele Progr'sprachen lassen sich einer der beiden Kategorien zuordnen. Ausnahmen sind hybride Sprachen (beide Kategorien).

Beispiel (plakativ)

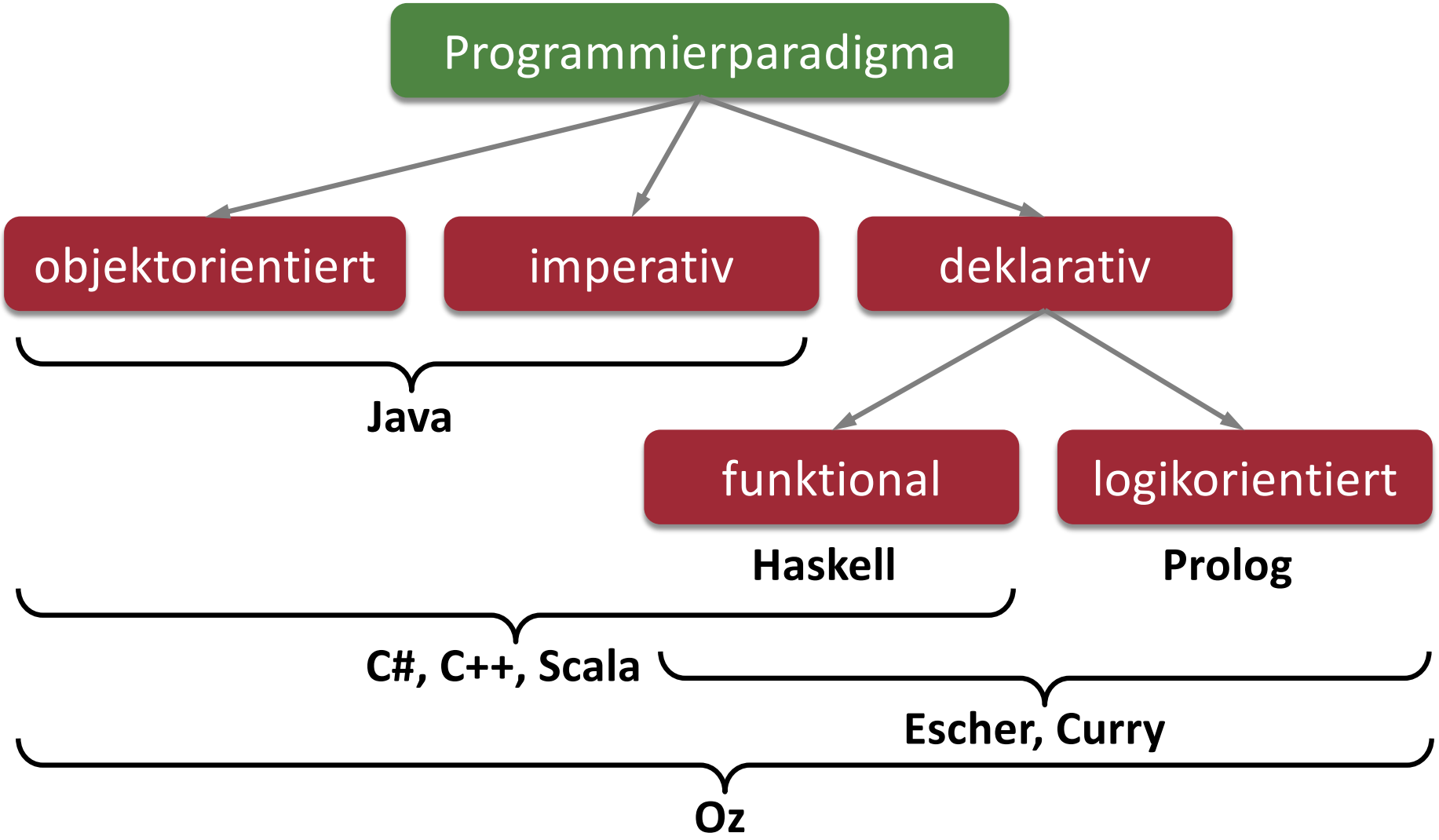


Nimm den Schlüsselbund vom Schlüsselbrett, geh zur Wohnungstür, öffne diese, schliesse diese beim Verlassen der Wohnung, geh die Treppe hinunter zur Haustür, öffne diese, schliesse diese beim Verlassen des Hauses, geh zum Auto, öffne die Autotür mit dem passenden Schlüssel, setzte dich auf den Fahrersitz, nimm den Zündschlüssel, ... usw. usf.

Im Kühlschrank ist keine Milch mehr.

Die Lösung ist ein Programm durch dessen Ausführung der Kühlschrank wieder mit frischer Milch gefüllt ist.

Relevante Subtypen



Weitere Dimensionen?

- Wikipedia* nennt derzeit 27 Top Level Paradigmen.
  - Event-driven
  - Data-driven
  - Agent-oriented
  - Parallel
  - Probabilistic
  - …

Nicht alle sind disjunkt/orthogonal.

* http://en.wikipedia.org/wiki/Programming_paradigm

Zwischenfazit

- Es gibt kein universelles Programmierparadigma.
- Die Menge der Programmiersprachenkonzepte ist endlich.
- Paradigmen haben konzeptbedingt bzw. situativ Vor- und Nachteile.
- Hat man sich ein Paradigma angeeignet, fällt das Umdenken auf ein anderes Paradigma oft schwer.
  - **Vorsicht**: Uninformierte neigen zu Subjektivität, Unsachlichkeit, bis hin zur Verblendung in Bezug auf ihr „Vorzugsparadigma“.
- Trend geht (seit Jahren) zu Multiparadigmen-Sprachen.

Deshalb sollten wir die wichtigsten kennen um:

- Systematisch beurteilen zu können wann (in welcher Situation) bzw. für welches Problem welches Paradigma besser geeignet wäre.