

Einführung – Teil I

HOW PROGRAMMING LANGUAGES PROLIFERATE:



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

Verbreitete Programmiersprachen

- Etwa 2500 Programmiersprachen bekannt¹
- TIOBE programming community index²
 - **Indikator** für Popularität einer Programmiersprache
 - Misst **nicht**
 - welche die *beste* ist,
 - oder welche die *meisten* lines of code (LOC) hat.
- Andere Indices
 - Language Popularity Index³
 - PYPL – PopularitY of Programming Language⁴

¹ <http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm> (per 17.02.20 nicht mehr abrufbar)

² <https://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

³ <http://lang-index.sourceforge.net/>

⁴ <https://pypl.github.io/PYPL.html>

TIOBE Top 20 Stand: 02/2020

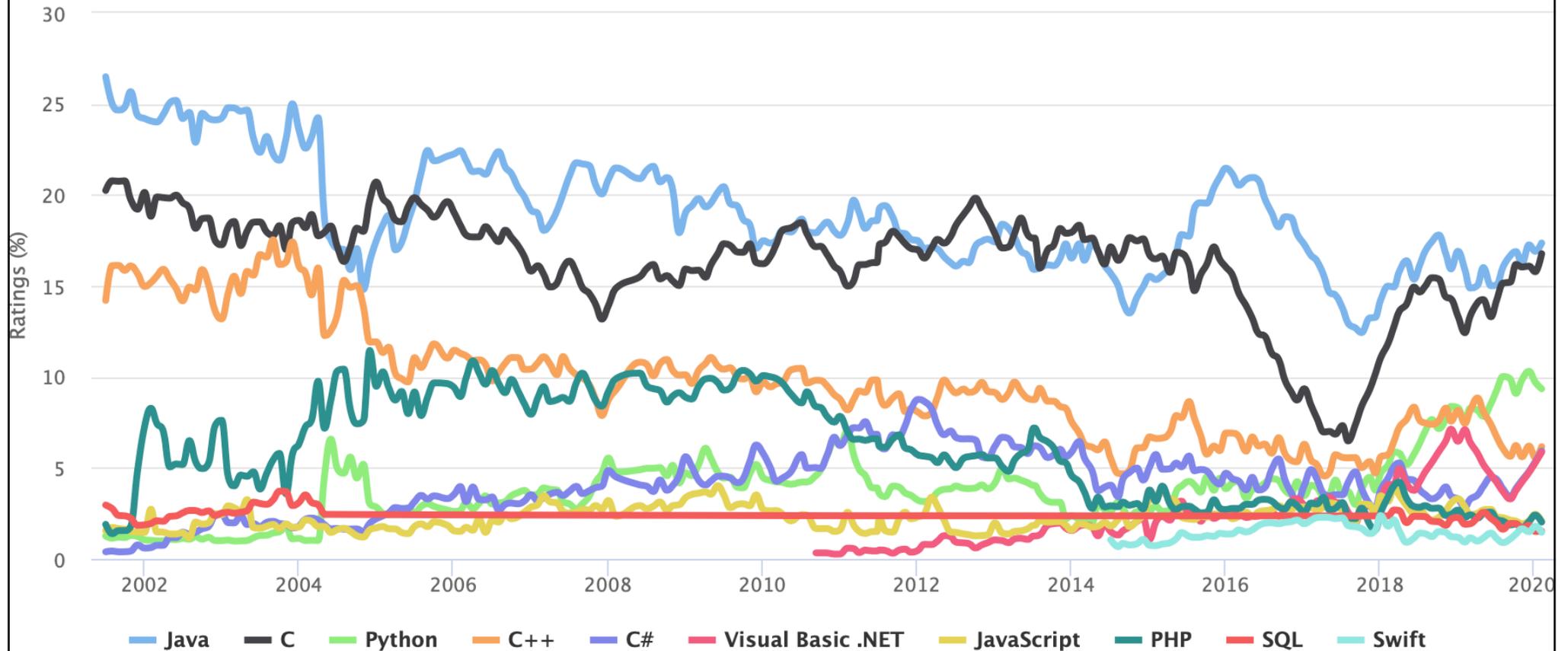
Feb 2020	Feb 2019	Change	Programming Language	Ratings	Change
1	1		Java	17.358%	+1.48%
2	2		C	16.766%	+4.34%
3	3		Python	9.345%	+1.77%
4	4		C++	6.164%	-1.28%
5	7	↑	C#	5.927%	+3.08%
6	5	↓	Visual Basic .NET	5.862%	-1.23%
7	6	↓	JavaScript	2.060%	-0.79%
8	8		PHP	2.018%	-0.25%
9	9		SQL	1.526%	-0.37%
10	20	↑↑	Swift	1.460%	+0.54%
11	18	↑↑	Go	1.131%	+0.17%
12	11	↓	Assembly language	1.111%	-0.27%
13	15	↑	R	1.005%	-0.04%
14	23	↑↑	D	0.917%	+0.28%
15	16	↑	Ruby	0.844%	-0.19%
16	12	↓↓	MATLAB	0.794%	-0.40%
17	21	↑↑	PL/SQL	0.764%	-0.05%
18	14	↓↓	Delphi/Object Pascal	0.748%	-0.32%
19	13	↓↓	Perl	0.697%	-0.40%
20	10	↓↓	Objective-C	0.688%	-0.76%

Scala? 25
TypeScript? 46
Haskell: 45 (39)
Prolog: >50 (35)

TIOBE - Langzeitverlauf

TIOBE Programming Community Index

Source: www.tiobe.com



Github-Programmiersprachenstatistik

<https://github.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 C die endliche Menge aller bekannten *Programmiersprachekonzepte*; z.B. *call-by-value*, *call-by-need*, *statisch typisiert*, *Operatorüberladung*, *referentielle Transparenz*, *strikte Auswertung*, *Pointer*, *Templates*, *Multiple Dispatch*, *strukturelle Typisierung* ...

Syntax, kognitive Aspekte, Vorlieben und prinzipbedingte Eigenschaften ausser Acht lassend, ist ein Programmierparadigma P eine nichtleere Teilmenge $P \subseteq C$.

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

Klassifikation von Progr'sprachen

Ziel (jeder Klassifikation):

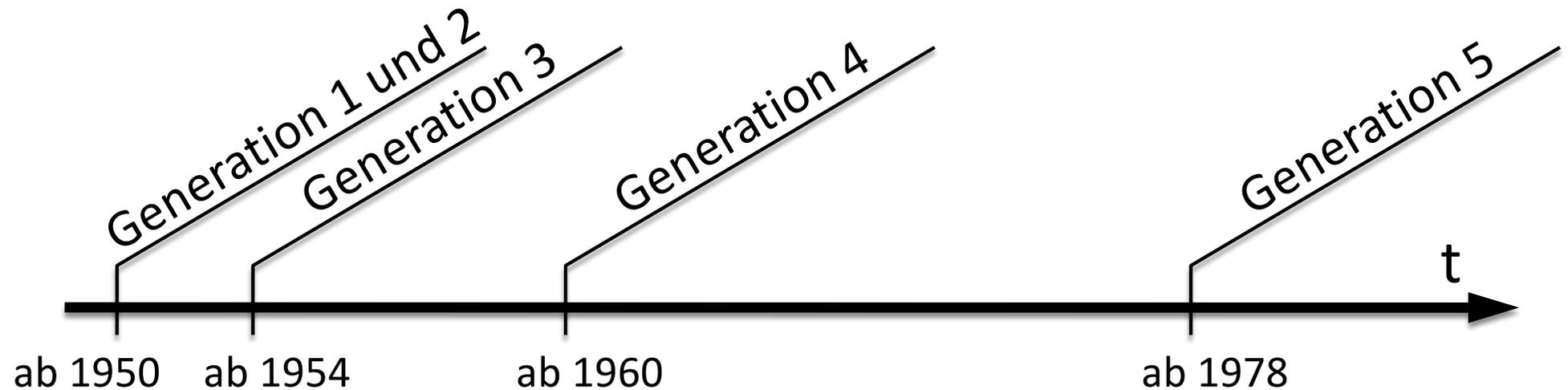
- Verstehen der Gemeinsamkeiten & Unterschiede
- Liefert:
 - Abgrenzung, Ordnung
 - Vokabular bzw. Terminologie – in unserem Fall sind dies **Programmiersprachen...**

...generationen (zeitlich)

...konzepte (ideell)

...klassen (kategorisch)

Klassifikation nach Generation



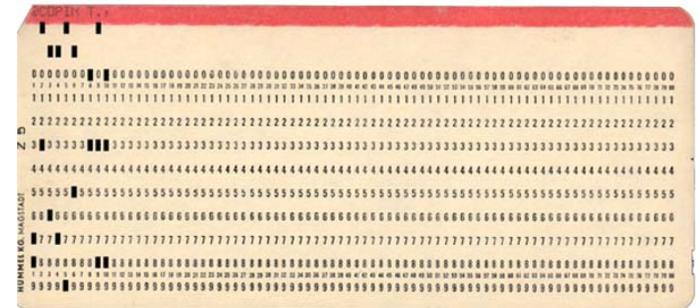
Grenzen unscharf

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:

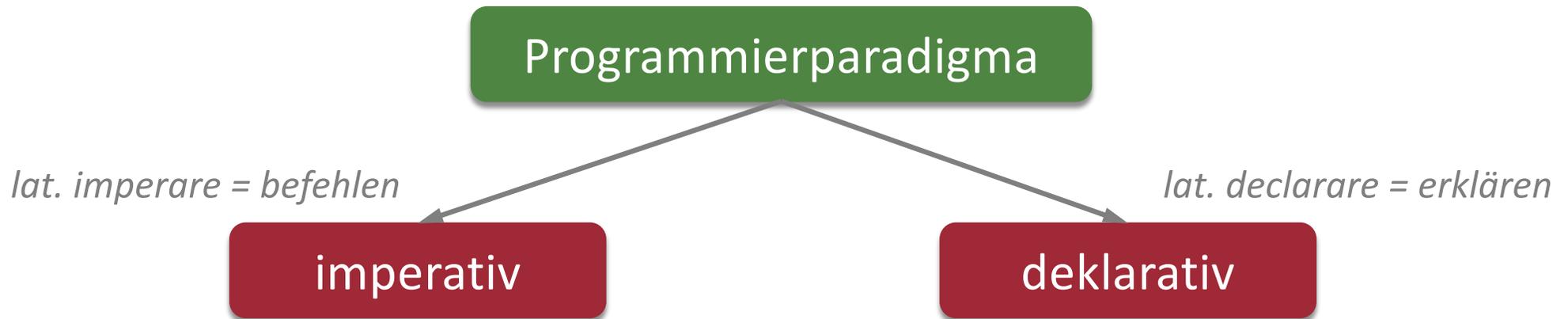
- Probleme werden durch Rand- bzw. Zwangsbedingungen implizit beschrieben
- System muss selbst eine Lösung finden/suchen unter Einhaltung der Bedingungen; siehe auch nachfolgende Klassifikation

Stammbaum

Einen chronologischen Überblick der wichtigsten Programmiersprachen liefert:

<https://www.levenez.com/lang/lang.pdf>

Klassifikation nach Prinzip



Programmierer beschreibt Ablauf

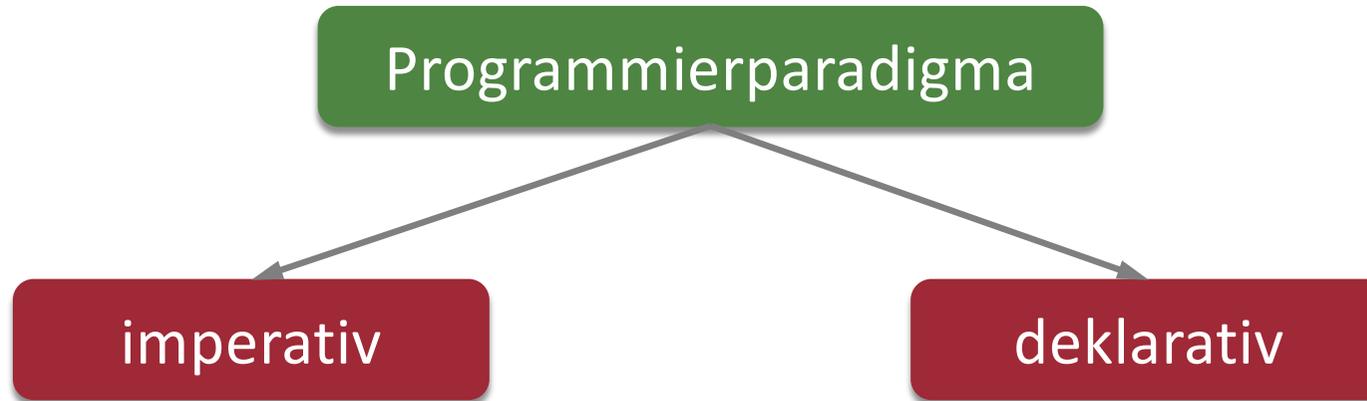
- **wie** ist das Problem zu lösen
- Schritt für Schritt
- mittels **Anweisungen** der Programmiersprache; nicht notwendigerweise der Maschine

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)

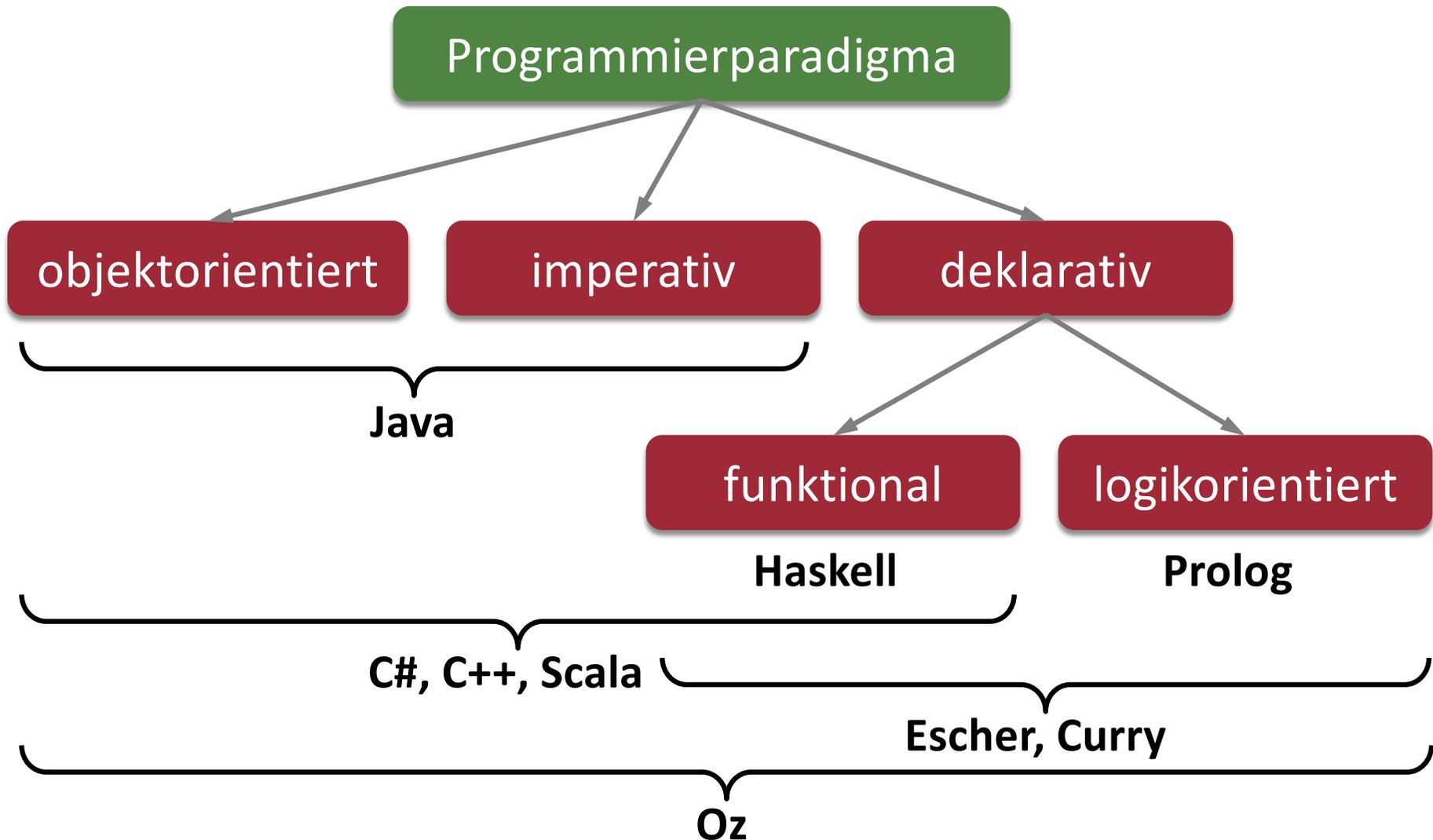


Nimm den Schlüsselbund vom Schlüsselbrett, geh zur Wohnungstür, öffne diese, schliesse diese beim Verlassen der Wohnung, geh die Treppe hinunter zur Haustür, öffne diese, schliesse diese beim Verlassen des Hauses, geh zum Auto, öffne die Autotür mit dem passenden Schlüssel, setze dich auf den Fahrersitz,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

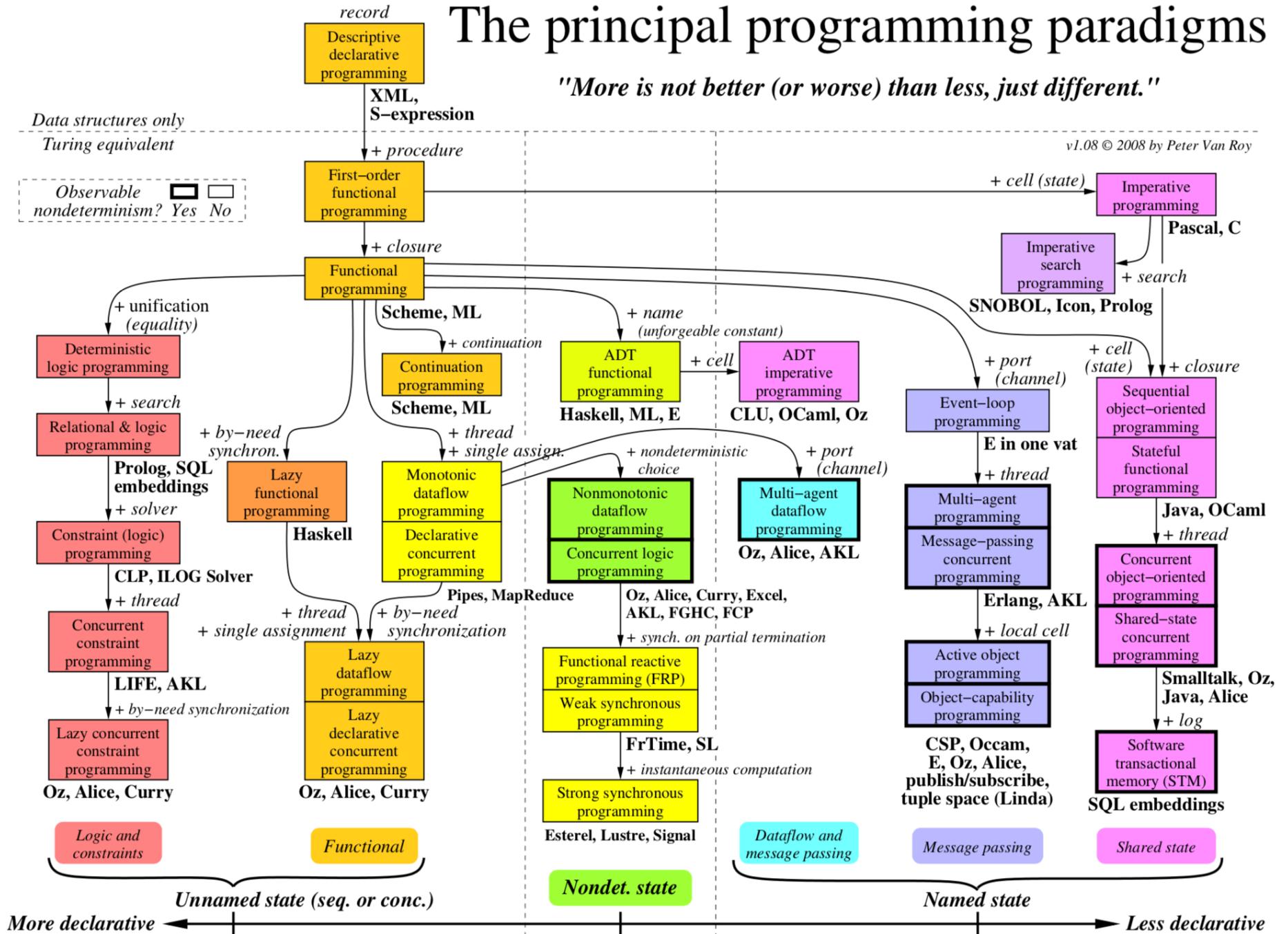
* http://en.wikipedia.org/wiki/Programming_paradigm



The principal programming paradigms

"More is not better (or worse) than less, just different."

v1.08 © 2008 by Peter Van Roy



Quelle: Peter van Roy, <https://www.info.ucl.ac.be/~pvr/paradigms.html>



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.