Programming Paradigms

1st Lecture

Prof. Janis Voigtländer

University of Duisburg-Essen

Summer Term 2017

To know another language is to have a second soul.
(Charlemagne, 747/748–814)

Who are we?

- Lecturer: Prof. Dr. Janis Voigtländer
  Room LF 259 (currently)

  Area: Formal Methods, Programming Languages

- Assistants: Dr. Tobias Hecking (`hecking@collide.info`)
  M.Sc. Sören Werneburg (`werneburg@...`)

- Tutor: Viet Hung Dinh

Who are you?

To my knowledge:

1. Students in Bachelor Progr. “Angewandte Informatik”
2. Students in Bachelor Progr. “Computer Engineering (ISE)”

Some lectures you have presumably attended

Angewandte Informatik:

- Grundlegende Programmiertechniken
- Fortgeschrittene Programmiertechniken
- maybe, Softwaretechnik

Computer Engineering (ISE):

- Fundamentals of Programming
- maybe, Object-oriented Programming or some of the above

This lecture

Weekly slot:

- Wednesday, 12:15–13:45, in LB 134
- 15 times
- Warning: I may sometimes arrive in a hurry.

Slides:

- . . . will be made available after each lecture.
- . . . will differ somewhat in style.

`http://moodle.collide.info/`

The exercises

Groups:

- Mon, 16:15–17:45, LE 105 (English)
- Tue, 10:15–11:45, LE 120 (German)
- Thu, 10:15–11:45, LC 137 (German)
- Fri, 08:15–09:45, LE 120 (German)

Starting next week, 24th April!

Some adjustments will be made for the free days/holidays.

`http://moodle.collide.info/`

Role of the exercises

There is no mandatory attendance or grading of solutions.

But to be successful in the course, doing the exercises is important!

In particular, resist the temptation of not doing programming tasks on your own, and to completion.

And do go to the exercise sessions to benefit from help and discussion there.

Material in the course cannot be learned by heart. It needs practice.

Eventually, the exam

There will be a written exam, from which the course grade is derived. We do not know the date yet.

Registration will be via the exam office.

Consequences of the change of lecturer

I am a different lecturer, with a different style.

The exam will be about the course as taught this term.

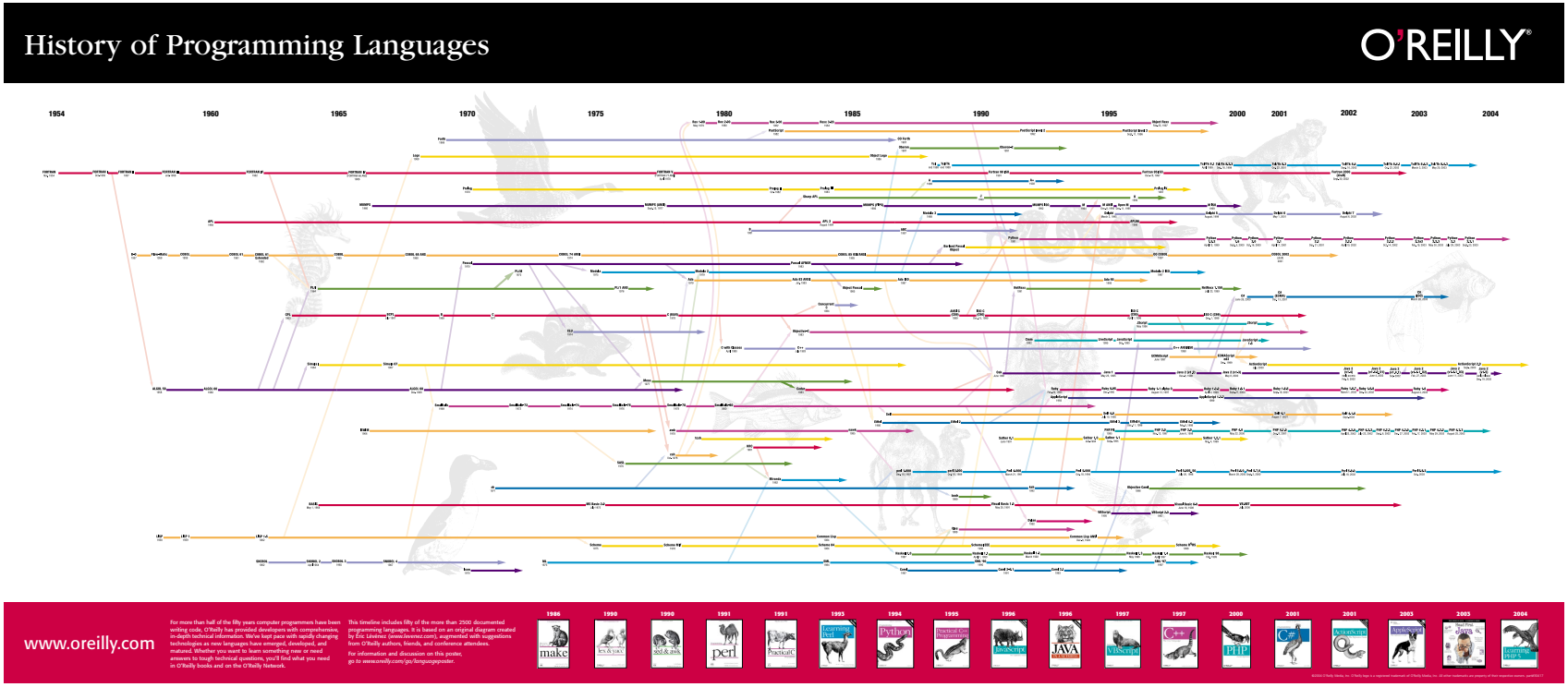
The course will have somewhat more emphasis on programming.

There will be less emphasis on object-orientation.

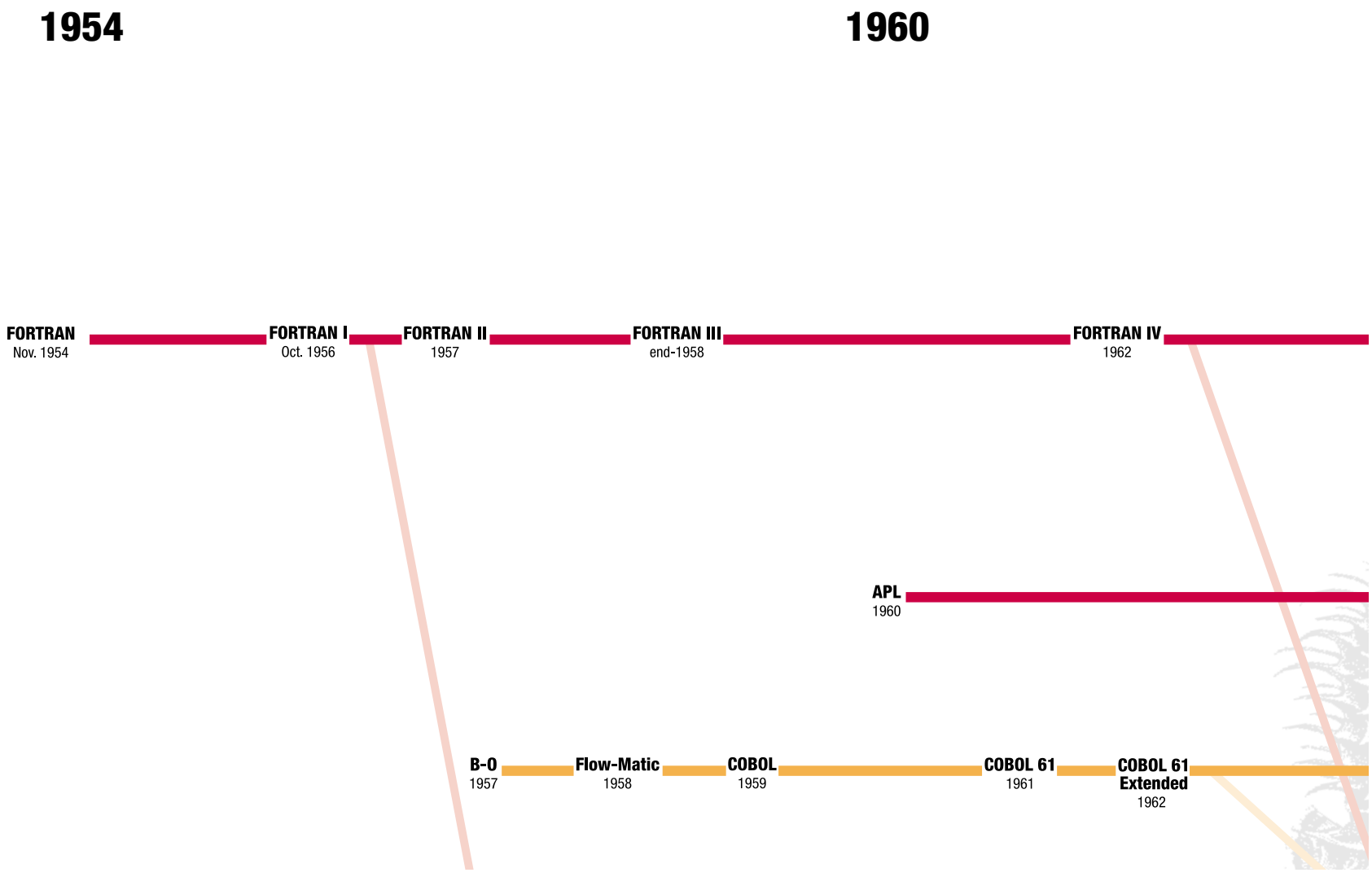
But the course assistants will provide a good degree of continuity!

Introduction

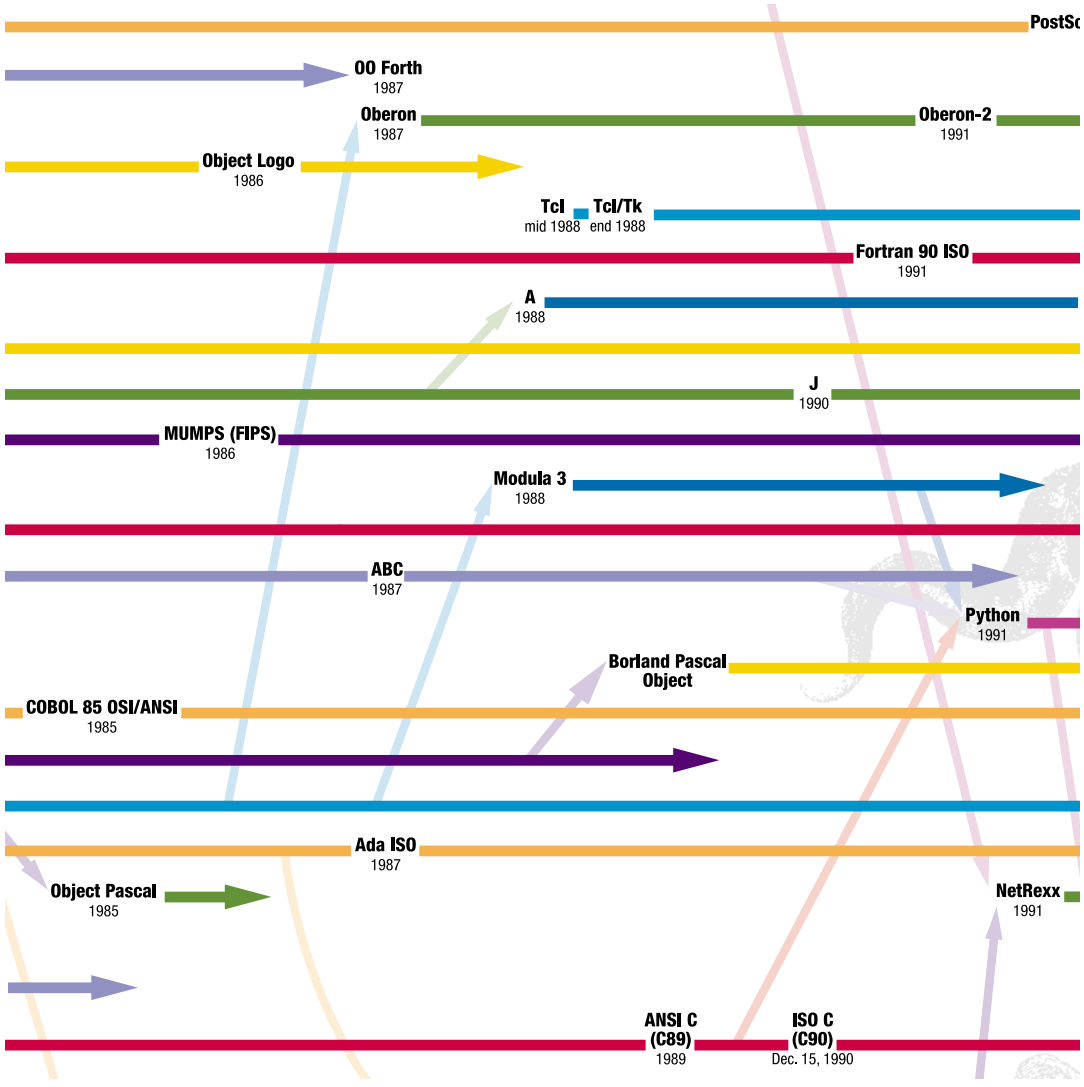
Many high-level programming languages exist



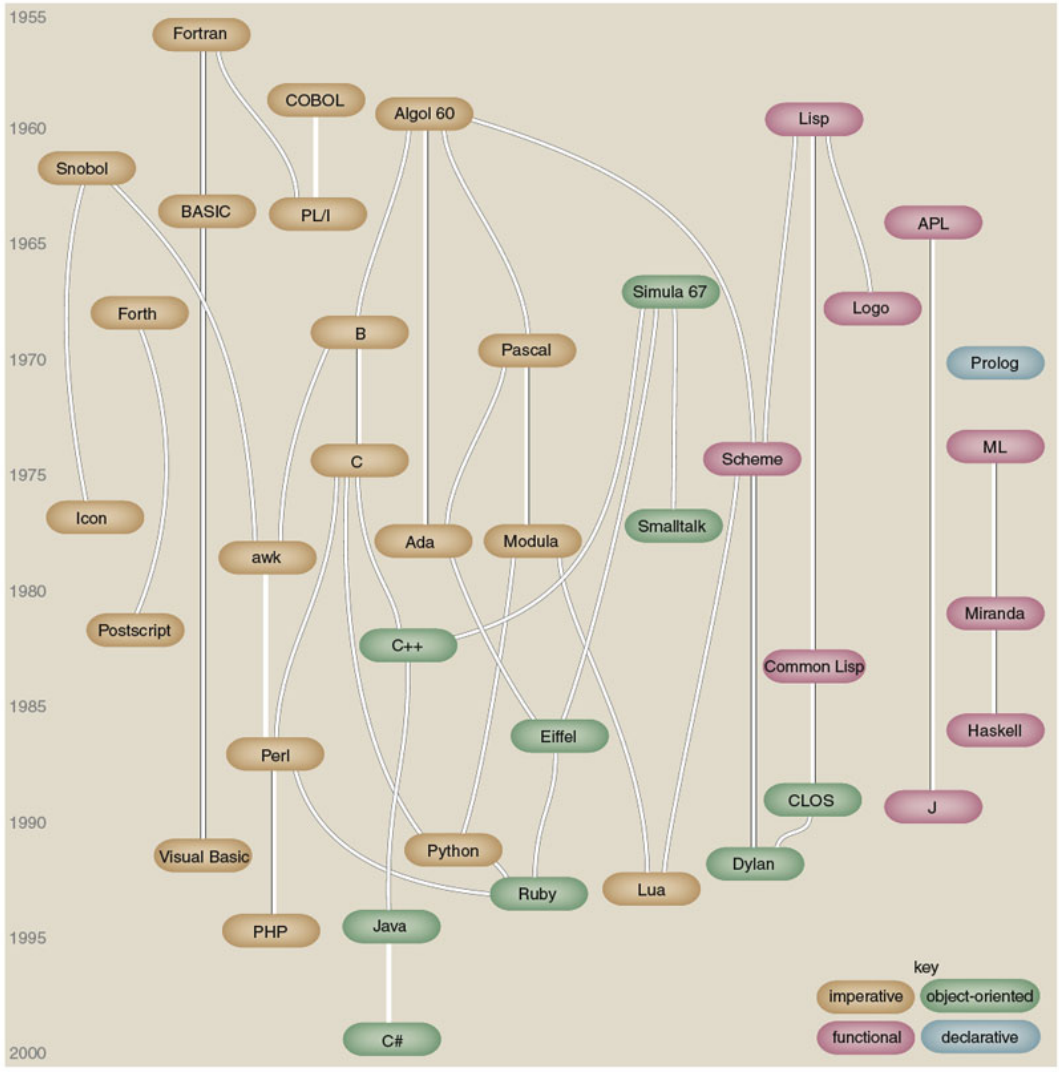
Many high-level programming languages exist



Many high-level programming languages exist

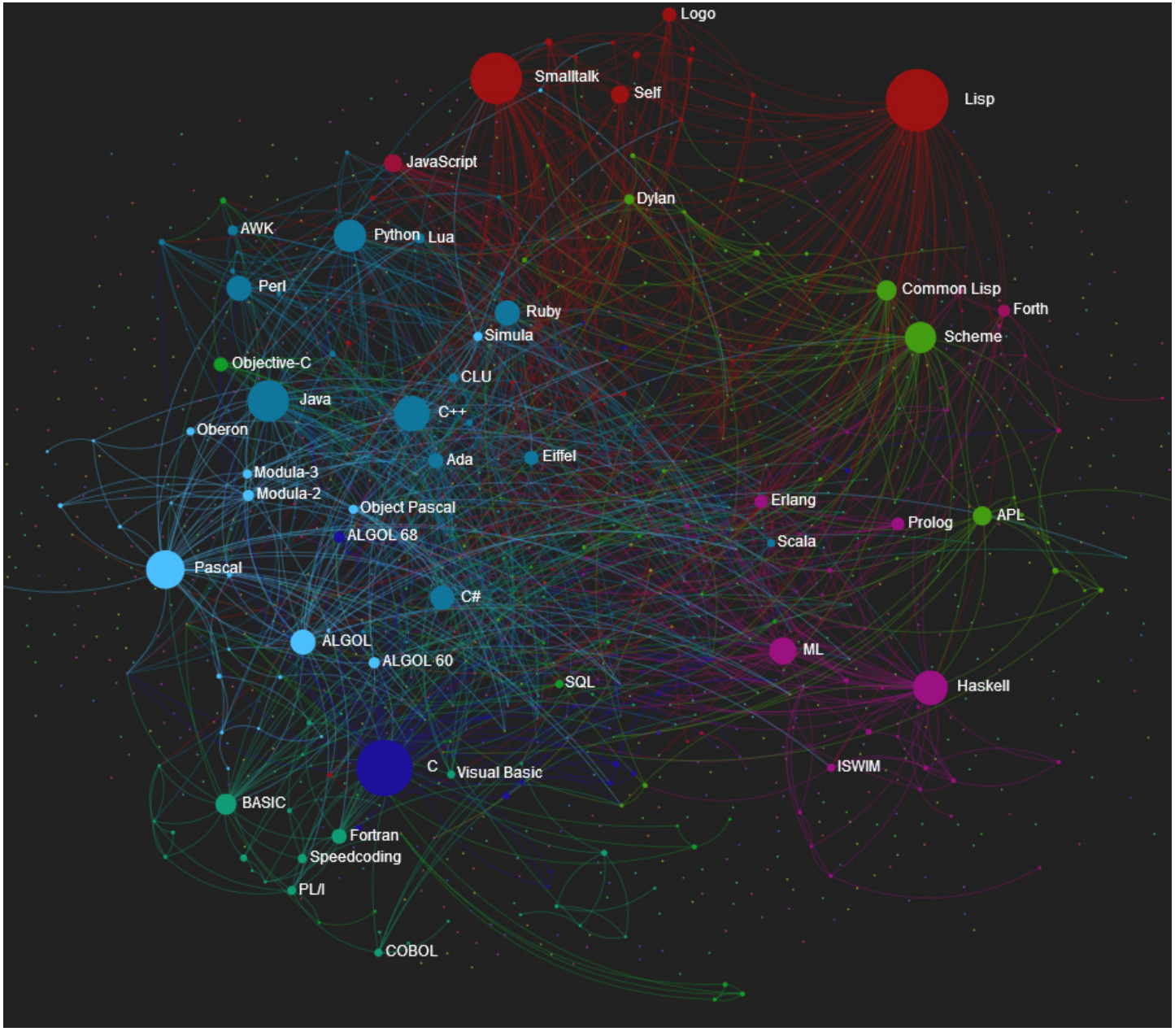


Another perspective



From „American Scientist“: The Semicolon Wars

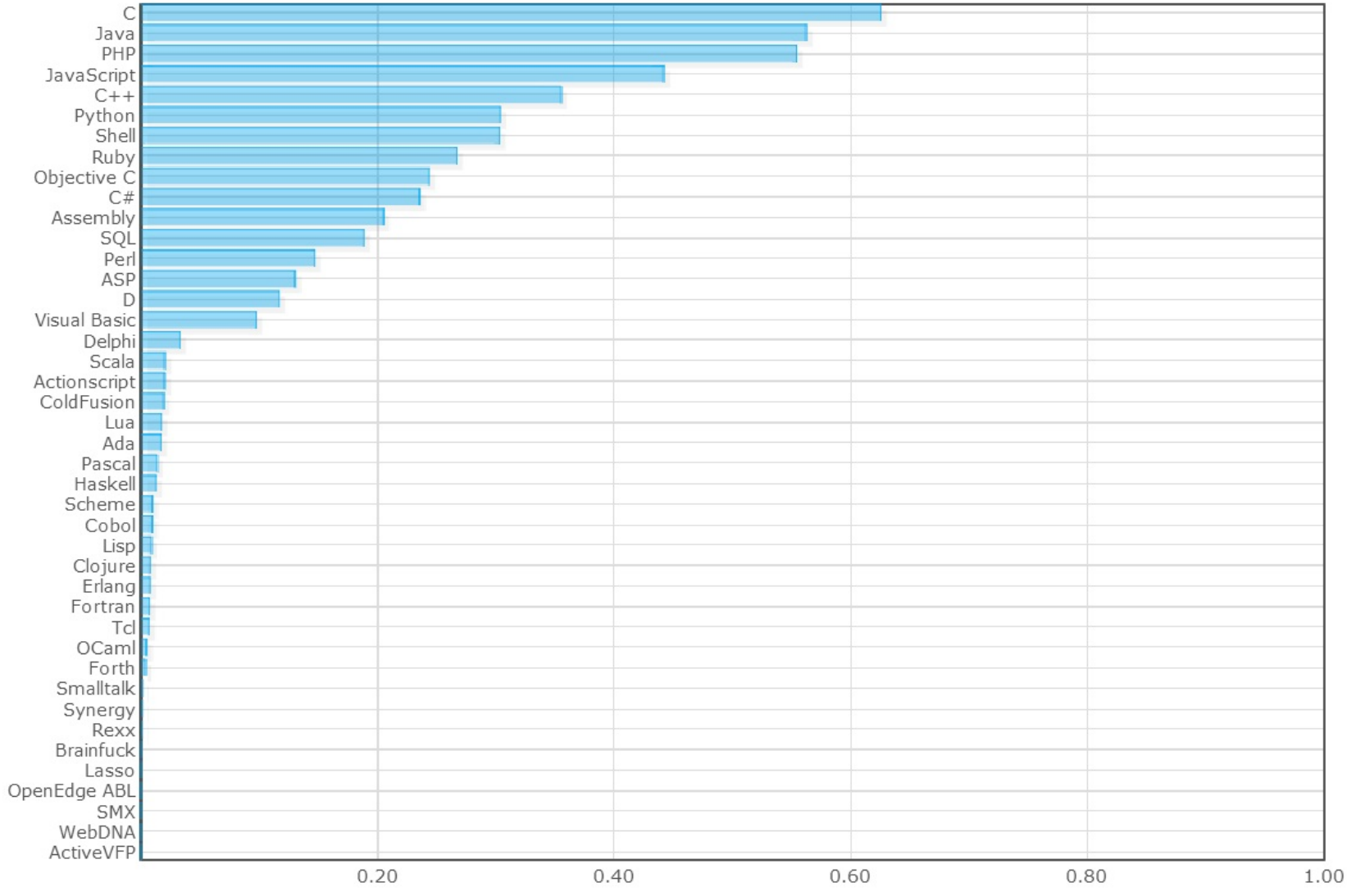
http://preview.tinyurl.com/language-influences



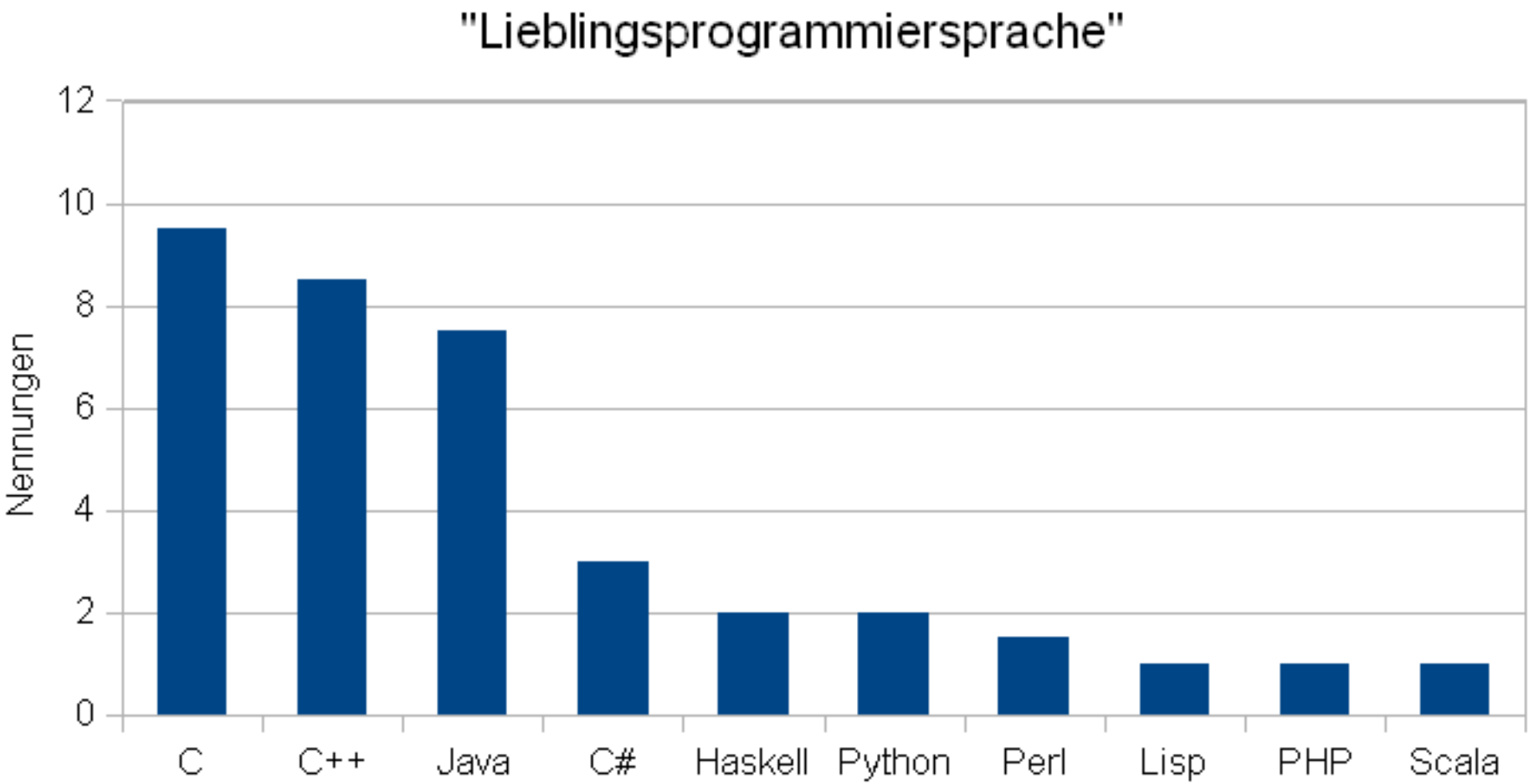
http://preview.tinyurl.com/popular-languages

Normalized Comparison

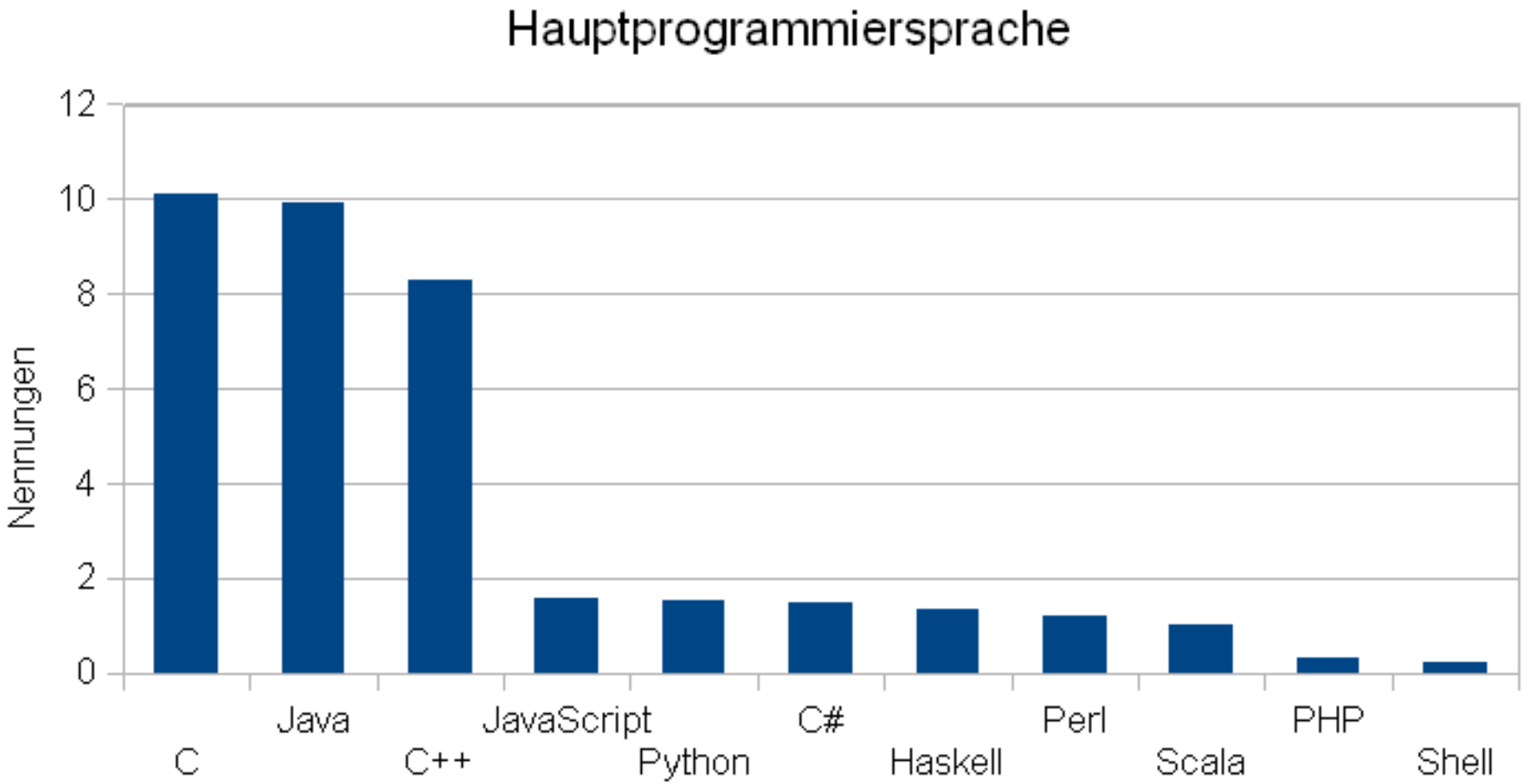
This is a chart showing combined results from all data sets, listed individually below.



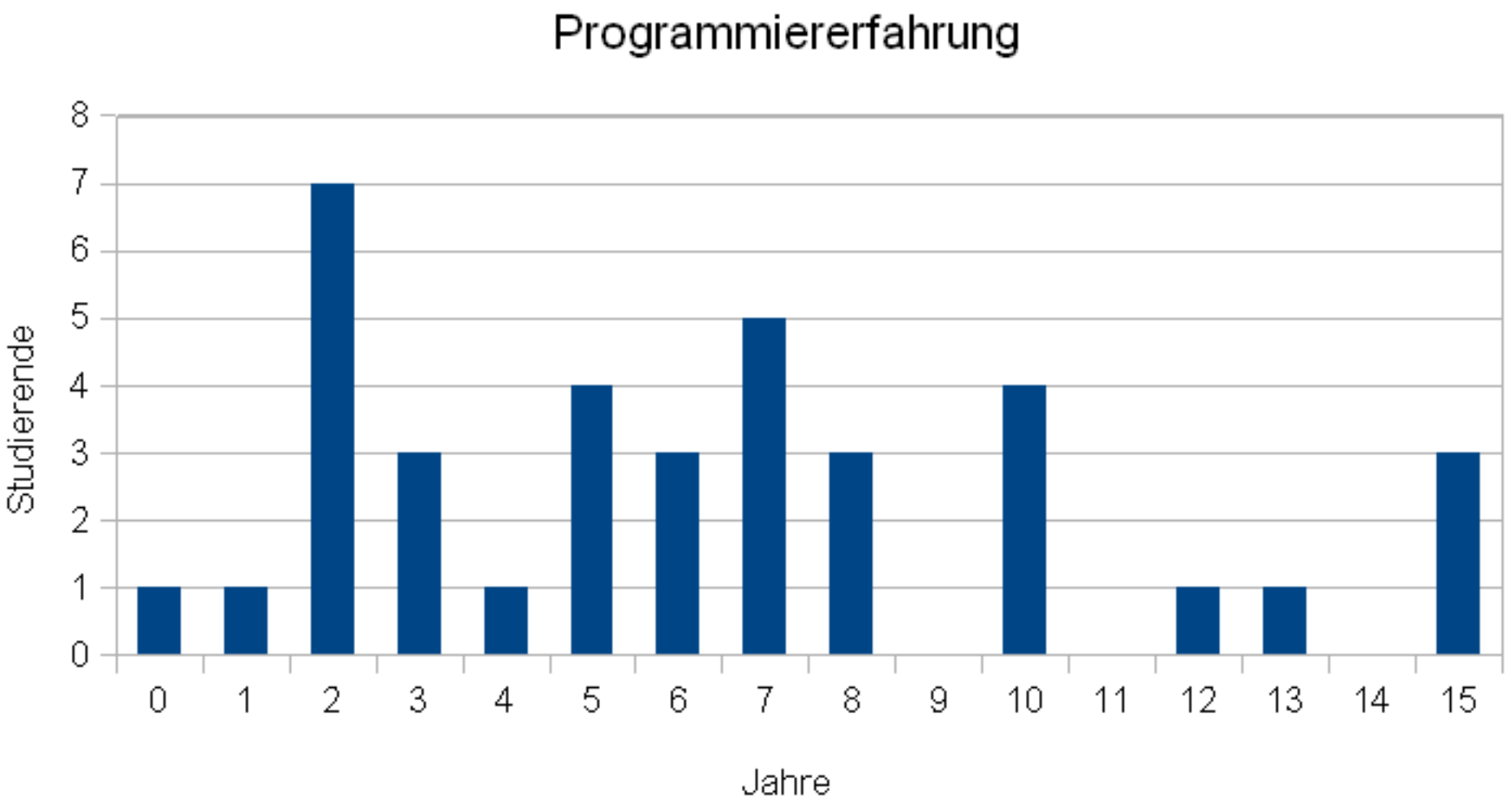
Survey at begin of a course a few years back (2013)



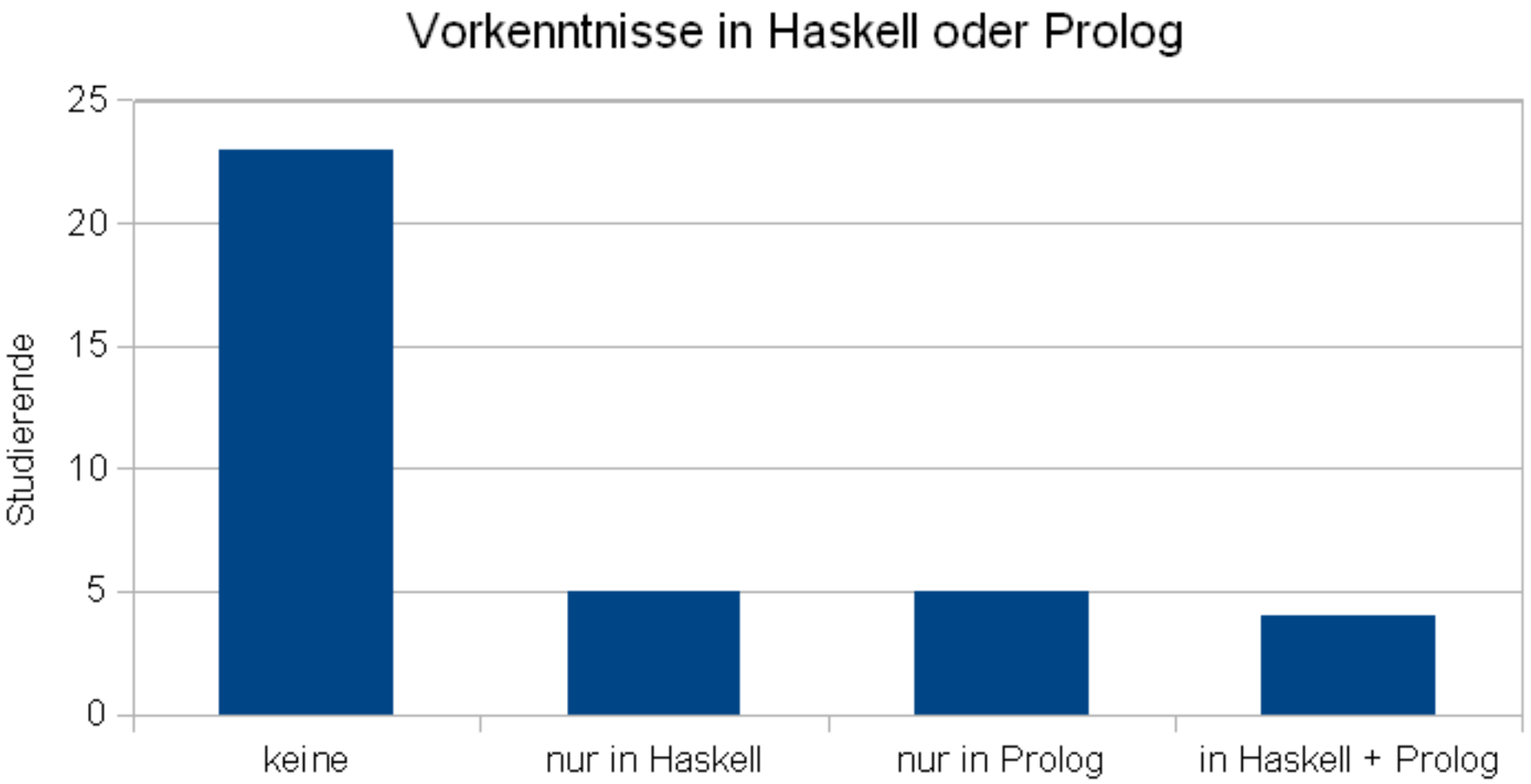
Survey at begin of a course a few years back (2013)



Survey at begin of a course a few years back (2013)

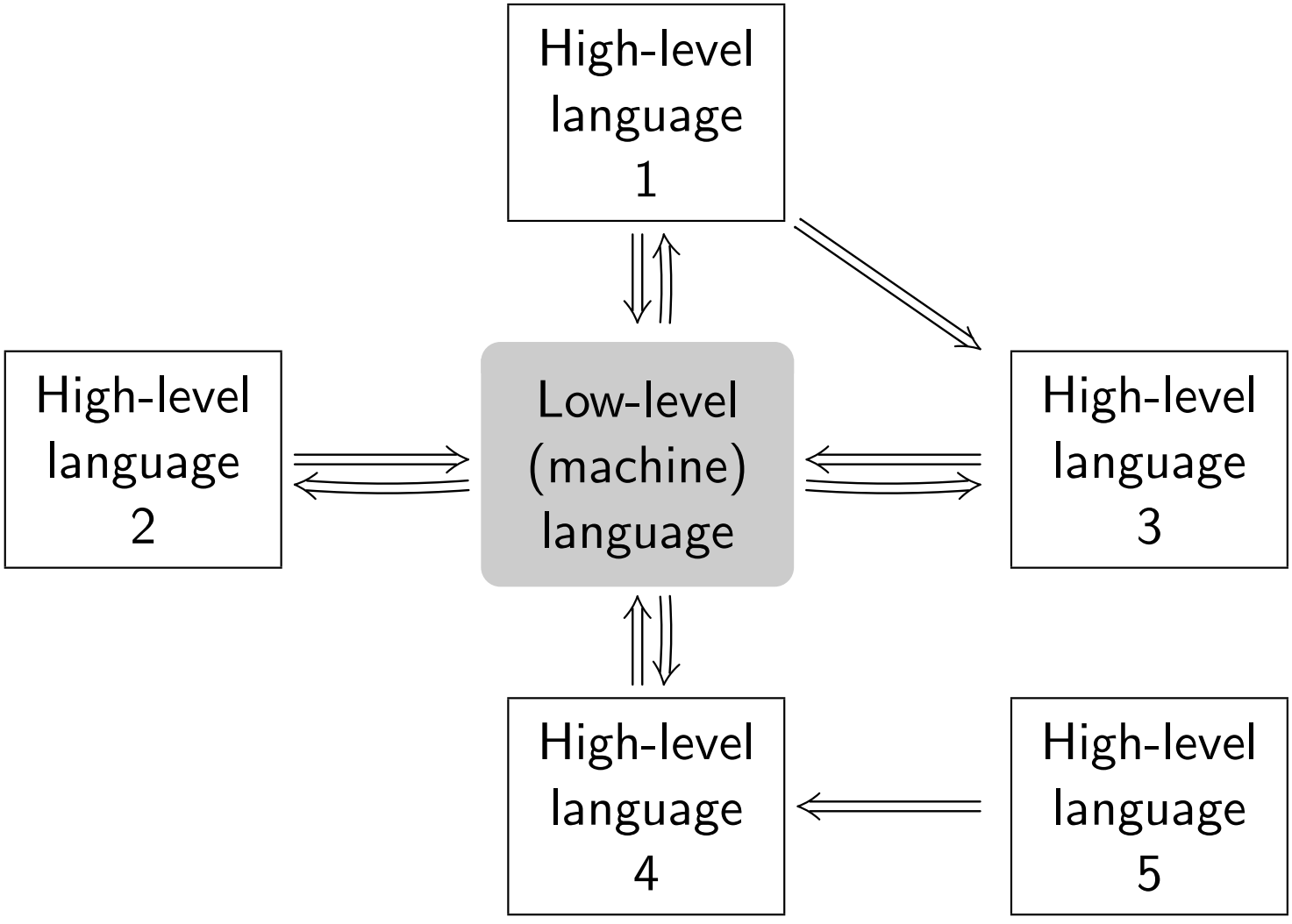


Survey at begin of a course a few years back (2013)



So, can one language do "more" than others?

Certainly not:



What, then, separates programming languages?

Some relevant distinctions:

- syntactically rich vs. syntactically scarce (e.g., APL vs. Lisp)
- verbosity vs. succinctness (e.g., COBOL vs. Haskell)
- compiled vs. interpreted (e.g., C vs. Perl)
- domain-specific vs. general purpose (e.g., SQL vs. Java)
- sequential vs. concurrent/parallel (e.g., JavaScript vs. Erlang)
- typed vs. untyped (e.g., Haskell vs. Prolog)
- dynamic vs. static (e.g., Ruby vs. ML)
- declarative vs. imperative (e.g., Prolog vs. C)
- object-oriented vs. ???
- ...

And, yet, there are common principles.

Approaches to the specification of languages

- . . . describing syntax.
- . . . describing semantics.

Implementation strategies.

Language concepts:

- variables and bindings
- type constructs
- control structures and abstractions

Paradigms that span a whole class of languages.

A rough plan of the lecture

- A look at specifying syntax of programming languages.
- A look at specifying semantics of imperative programming languages.
- Functional programming using Haskell.
- Evaluation of functional programs.
- Typing concepts like inference, genericity, polymorphism.
- Logic programming using Prolog.
- Comparisons of concepts like variables, statements vs. expressions, etc. in the different languages.
- Potentially, outlook to other paradigms, functional-logic, or influence of functional on object-oriented, ...