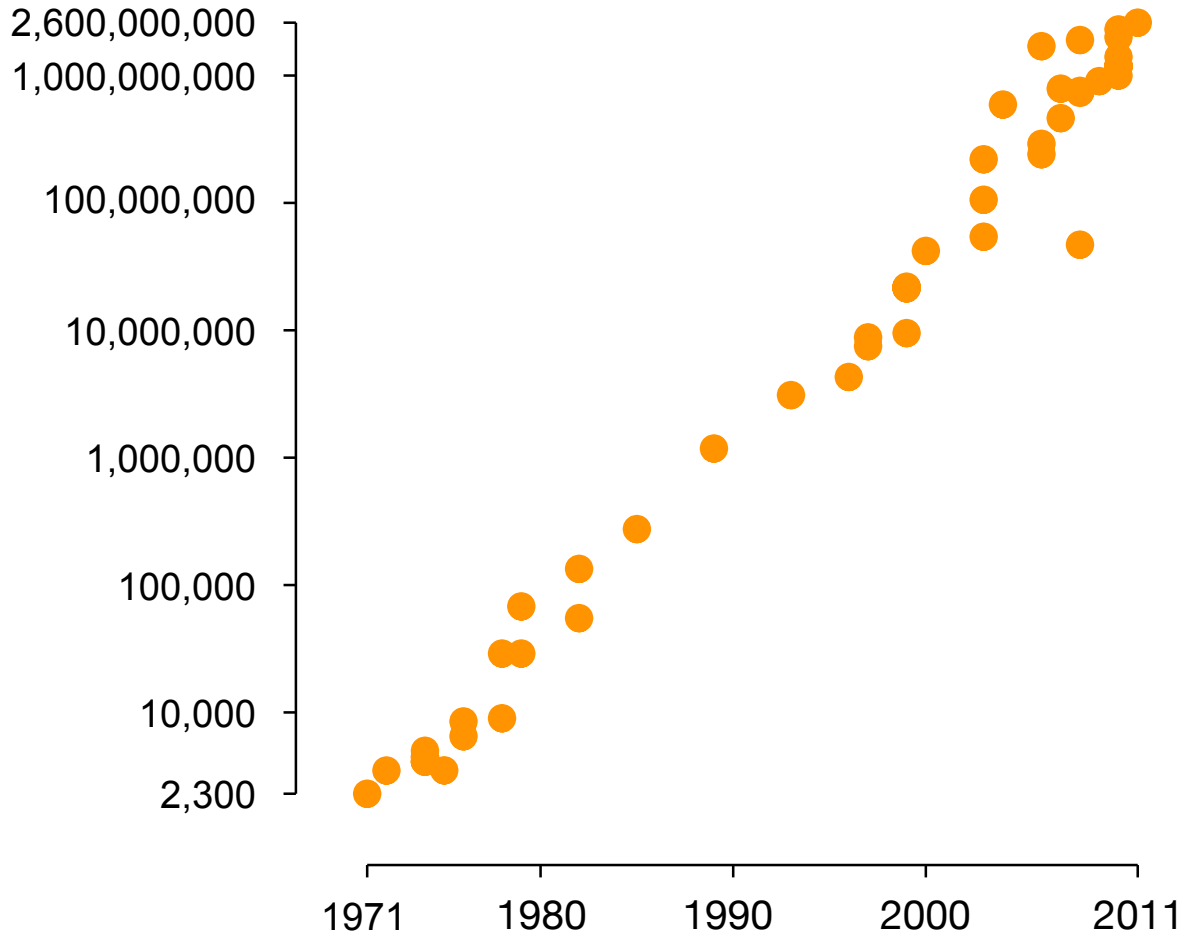
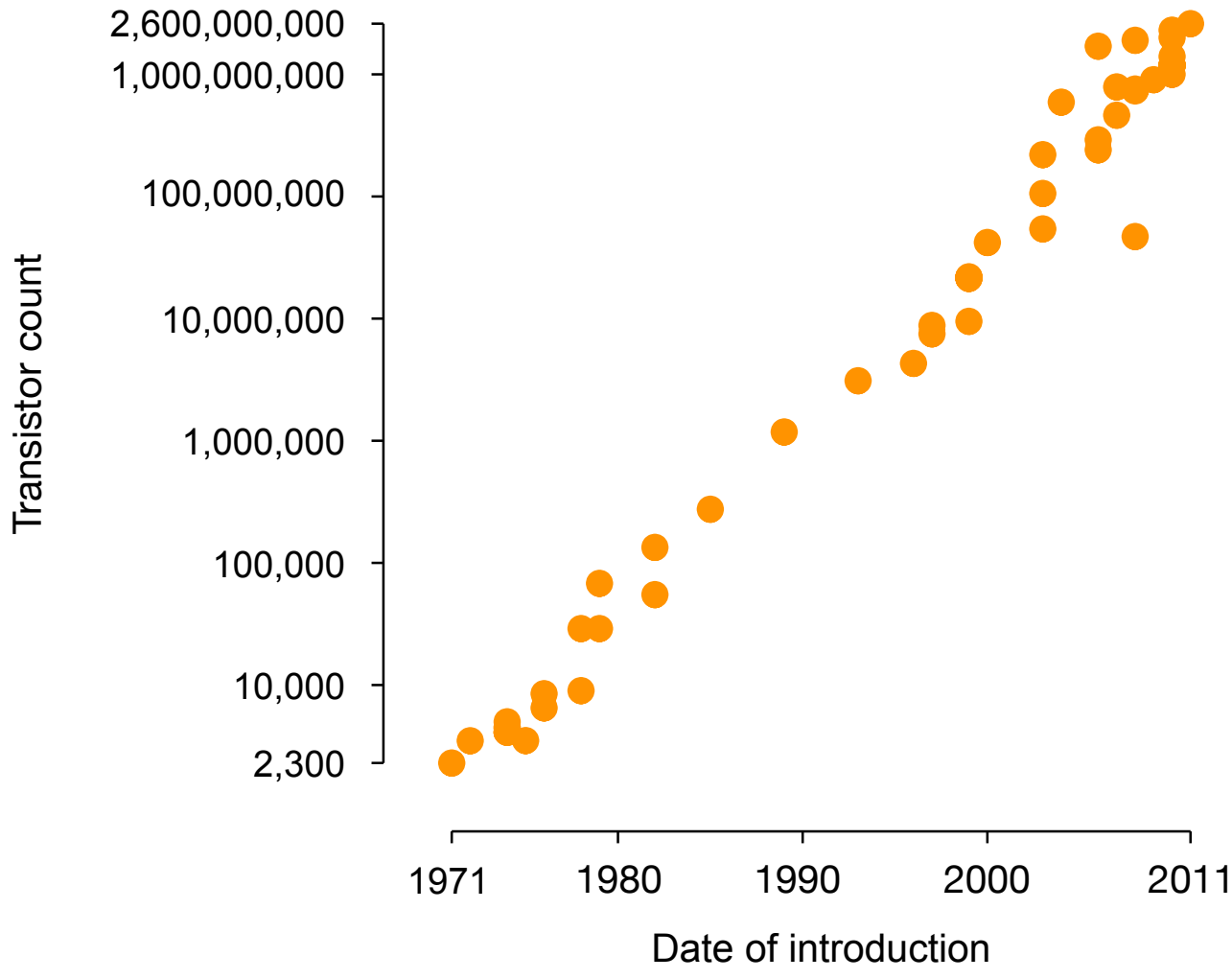
WAS WIRD HIER GEZEIGT?



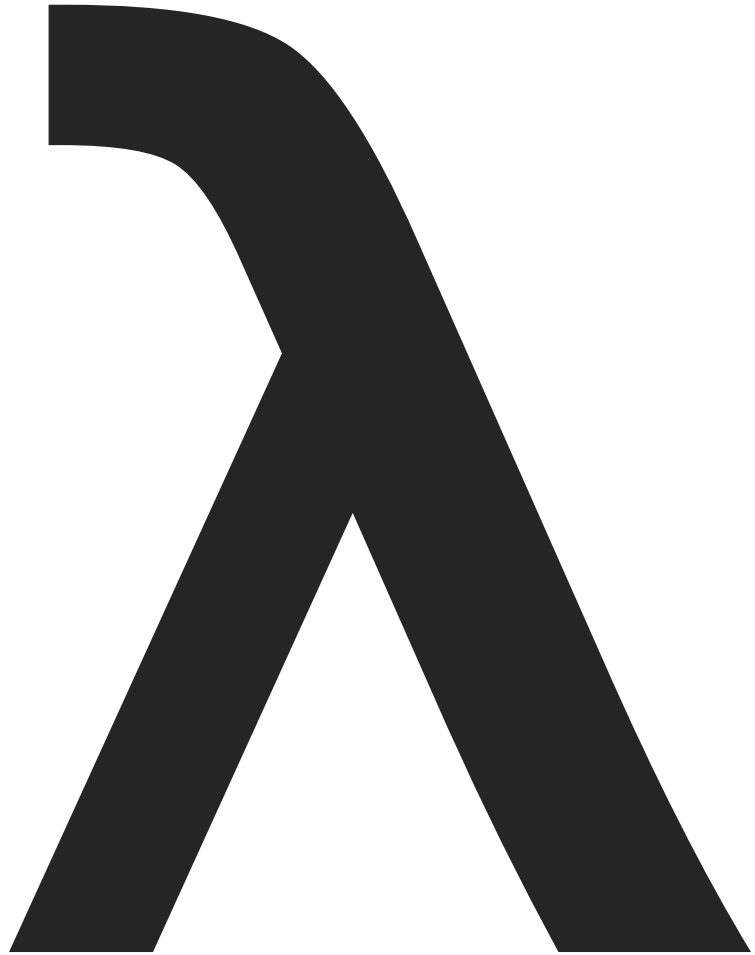
TRANSISTOR COUNTS 1971-2011 & MOORE'S LAW



FUNKTIONALE PROGRAMIERUNG

Nebenläufigkeit & Parallelisierung

Seminar, WS2016



Jan-Philipp Willem

Prof. Dr. Sandro Leuchter
Fakultät für Informatik
Hochschule Mannheim

GLIEDERUNG

1. Nebenläufigkeit / Parallelisierung
2. Threads / Locking
3. Functional Paradigm 101
4. Elixir
5. Fazit

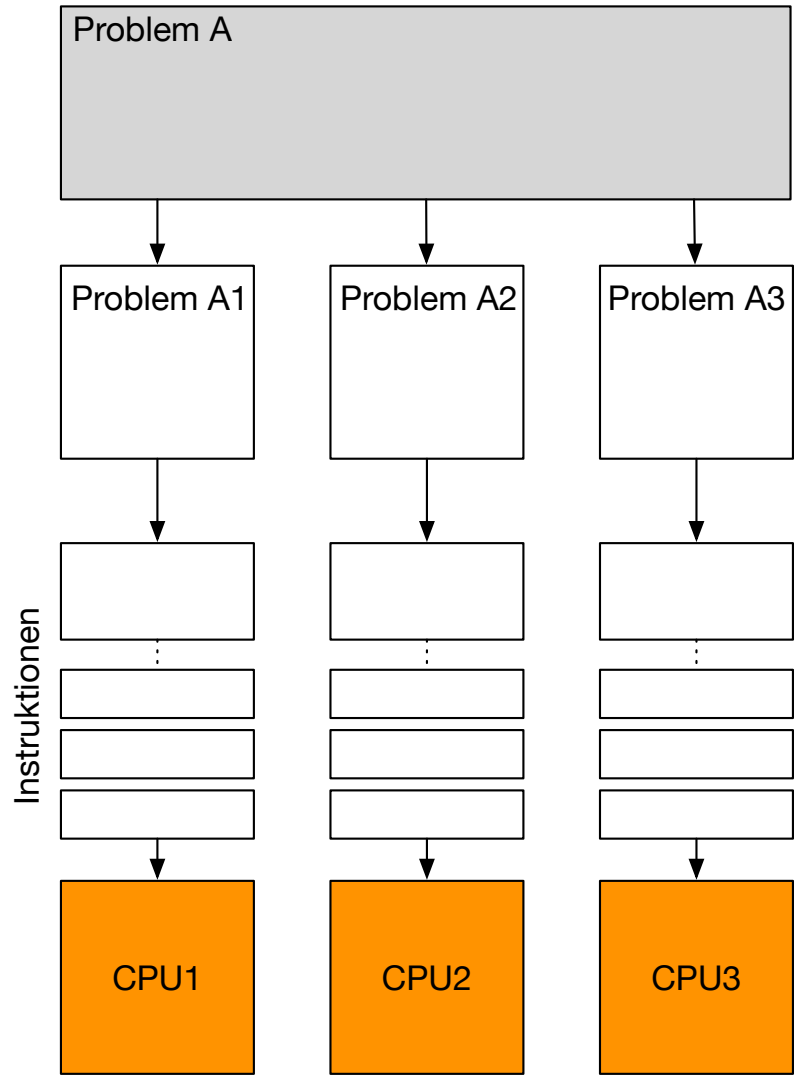
NEBENLÄUFIGKEIT ODER PARALLELISIERUNG?

PARALLEL

→ Synonyme:
`nebeneinander`,
`nebenläufig`

→ Informatik:
parallel ≠ nebenläufig !

→ „schneller als sequenzielles Programm, durch **gleichzeitiges** Ausführen von Anweisungen“



NEBENLÄUFIG

→ Concurrent (engl.)

ROB PIKE - 'CONCURRENCY IS NOT PARALLELISM'

→ „Concurrency is about dealing with lots of things at once.“

→ „Parallelism is about doing lots of things at once.“

→ „Concurrency is about structure, parallelism is about execution.“

THREADS / LOCKING

WAS IST EIN THREAD?

→ Punkt 1

→ Punkt 2

HÄUFIGE BUGS

Race-Condition

→ ++

→ ++

Deadlock

→ –

→ –

FAZIT: THREADS PROGRAMMING

→ SState is Evil”

→ Viele Sprachen besitzen threads als feature,

→ wenige Sprachen helfen mit Tooling oder Abstraktion dem Programmierer selbst!

FUNCTIONAL PARADIGM 101

FUNCTIONAL PARADIGM 101

→ reine Funktionale Sprachen
→ imutable // mutable
→ no side-effects
→ deterministic
→ data-in <-> data-out
→ functions as first-class citizens
→ lamdas

ELIXIR

ELIXIR

→ moderne Variante von Erlang (1987, Ericsson)
→ Beam-VM
→ Fault-Taulerant
→ „Let it crash“
→ Supervision-Trees
→ Shared & Distributed Memory
→ Open Telecom Platform (OTP)

OTP / ACTOR-MODEL

- → Punkt 1
- → Alternativen:
  - → Akka (Java/Scala)
  - → Akka.NET
  - → Pykka (Python)
  - → CAF (C++)
  - → Celluloid (Ruby)

Img-Exp

LIST-PROCESSING IN ELIXIR: MAP

LIST-PROCESSING IN ELIXIR: REDUCE

LIST-PROCESSING IN ELIXIR: FILTER

ELIXIR-STREAMS

FAZIT

FAZIT: FUNCTIONAL PROGRAMMING

Vorteile

→ Punkt 1

→ Punkt 2

Nachteile

→ Punkt 1

→ Punkt 2

FUNCTIONAL STYLE IN IMPERATIVE LANGUAGES