

Außergewöhnliche Features in unüblichen Sprachen

Prof. Dr. Matthias Tichy¹, Stefan Kögel¹

¹Institute of Software Engineering and Compiler Construction
Ulm University

Ulm, 15.10.2015

Introduction

- Seminar (Bachelor & Master)
- Supervisor: Stefan
- Interesting languages/topics that lack mainstream attention
- State of the art (known since 1970)

Requirements

- Curiosity
- Independence
- Programming knowledge
- English skills

Goals

Learn to:

- Think outside the box
- Read papers/do research
- Write a scientific work
- Hold a presentation
- *Use arcane technology to solve real world problems*

What you need to do

- Keep the deadlines
- Write a good essay (10-12 pages)
- Presentation (20 minutes) + Discussion (10 minutes)
- Participation
- Don't plagiarise

Contents of your essay

- Motivation
- Explanation
- Example
- Research/Literature (at least three papers or books)
- *Applications?*
- English is preferred, if possible (Please use a spell checker)

Important Dates & Deadlines

- 15.10. Einführungsveranstaltung (heute)
- 21.10. Vortrag Wissenschaftliches (aus)arbeiten
- 8.11. **Deadline** Gliederung & Quellen
- 9.-13.11. Besprechung Gliederung & Quellen
- 13.12. **Deadline** Abgabe 1te Version & Peer-Reviewstart
- 20.12. **Deadline** Abgabe Peer-Reviews
- 13.01. Vortrag Präsentieren für Dummies
- 17.01. **Deadline** Abgabe verbesserte Version
- 18.-22.01. Besprechung verbesserte Version
- 7.02. **Deadline Abgabe finale Version**
- 8.-12.02. **Präsentationen**

Erlang

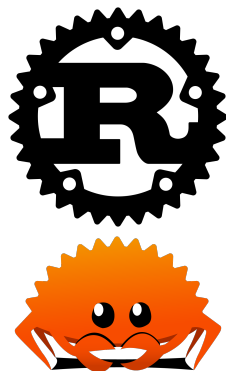
- > Concurrency oriented programming
- > Supervision trees
- > Let it crash

- > How does concurrency work?
- > How are errors/failures handled?
- > Best practices?
- > Differences to other languages/platforms?



Cyclone/Rust

- ; Systems programming language developed by Mozilla
 - ; Memory safe replacement for C++?
 - ; Regions/lifetimes
-
- ; What are regions/lifetimes?
 - ; How is memory safety accomplished? Any trade-offs?
 - ; No more segfaults?
 - ; How does this interact with concurrency?



Agda

- λ Dependent types
- λ Lift values to the type level
- λ You should know some Haskell!
- λ What are dependent types and how do they work?
- λ No more `ArrayIndexOutOfBounds`?
- λ How do you work with Agda?
- λ Applications outside of academia?

```
head : {A : Set}{n : Nat} -> Vec A (suc n) -> A
head (x :: xs) = x
```



<http://learnyouanagda.liamoc.net/>

Clojure/Scala

- () Immutable data structures
- () Concurrency made simple, not easy
- () How do those structures work?
- () What are the trade-offs?
- () How are they used in practice?
(Especially in concurrent settings)



λ Meta programming

λ What is it good for?

λ How does it work?

λ How does it interact with type safety?

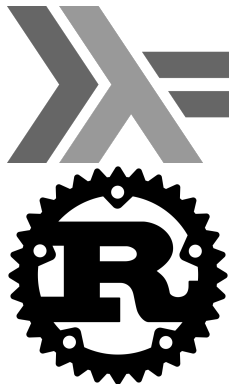
λ Example applications?



Wait, haven't I already heard of this? Wasn't this invented in like 1970 by a language wit **L**ots of **I**rritating and **S**uperfluous **P**arenthesis?

Haskell/Rust

- λ Algebraic data types
 - λ Making illegal states unrepresentable
-
- ; How does it work?
 - ; No more `NullPointerException`?
 - ; Example applications? Solved Problems?
 - ; Usage in other languages/designs?



Java Modelling Language

@ Contracts

@ How does it work?

@ What kind of problems does it catch?
Which does it miss?

@ Real world applications?



```
/*@ public normal_behavior
    @   requires y >= 0;
    @   ensures \result * \result <= y
    @           && y < (Math.abs(\result) + 1)
    @           * (Math.abs(\result) + 1);
    @*/
public static int isqrt(int y)
{
    return (int) Math.sqrt(y);
}
```

Topics

Erlang	concurrency & error handling
Cyclone/Rust	memory safety (regions/lifetimes)
Agda	dependent types
Clojure/Scala	immutable data structures
Haskell/Lisp	meta programming
Haskell/Rust	type system
Java Modelling Language	contracts

End

Thanks for coming.

Any questions left?