Außergewöhnliche Features in unüblichen Sprachen

Prof. Dr. Matthias Tichy\textsuperscript{1}, Stefan Kögel\textsuperscript{1}

\textsuperscript{1}Institute 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-Reviewerstart
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

- Dependent types
- Lift values to the type level
- You should know some Haskell!

- What are dependent types and how do they work?
- No more ArrayIndexOutOfBounds?
- How do you work with Agda?
- Applications outside of academia?

head : {A : Set}{n : Nat} → Vec A (suc n) → A
head (x :: xs) = x

http://learnyouanagda.liamoc.net/
(1) Immutable data structures
(2) Concurrency made simple, not easy

(1) How do those structures work?
(2) What are the trade-offs?
(3) How are they used in practice?
   (Especially in concurrent settings)
Meta programming

What is it good for?
How does it work?
How does it interact with type safety?
Example applications?

Wait, haven’t I already heard of this? Wasn’t this invented in like 1970 by a language wit Lots of Irritating and Superfluous Parenthesis?
Haskell/Rust

- Algebraic data types
- Making illegal states unrepresentable

- How does it work?
- No more NullPointerExceptions?
- 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

<table>
<thead>
<tr>
<th>Language</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erlang</td>
<td>concurrency &amp; error handling</td>
</tr>
<tr>
<td>Cyclone/Rust</td>
<td>memory safety (regions/lifetimes)</td>
</tr>
<tr>
<td>Agda</td>
<td>dependent types</td>
</tr>
<tr>
<td>Clojure/Scala</td>
<td>immutable data structures</td>
</tr>
<tr>
<td>Haskell/Lisp</td>
<td>meta programming</td>
</tr>
<tr>
<td>Haskell/Rust</td>
<td>type system</td>
</tr>
<tr>
<td>Java Modelling Language</td>
<td>contracts</td>
</tr>
</tbody>
</table>
Thanks for coming.

Any questions left?