Learning Programming with Erlang or Learning Erlang with Ladybirds

Frank Huch

Christian-Albrechts-University of Kiel

15. Oktober 2007
students in computer science in Germany (also Kiel): less than 20% girls
computer science not taught in every school

Our approach for improvement:

- One week course introducing computer science
- Talks of research groups, information about studying computer science, discussion with students and female computer scientist from industry, a trip to an IT company, and
- a programming course with a final project.
- Exercises in a nice programming environment
- Zurich: Programming environment Kara
- Solve tasks by programming a little ladybird
• Exercises in a nice programming environment.
• Zurich: Programming environment Kara.
• Solve tasks by programming a little ladybird.

Advantages: Attractive task, good identification.

Programming with: Finite Automata, Java.

Our approach: Erlang.
• Exercises in a nice programming environment
• Zurich: Programming environment Kara
• Solve tasks by programming a little ladybird
• Advantages: Attractive task, good identification
Teaching Programming

- Excercises in a nice programming environment.
- Zurich: Programming environment Kara.
- Solve tasks by programming a little ladybird.
- Advantages: Attractive task, good identification.

Programming with: Finite Automata, Java
Our approach: Erlang

Frank Huch (CAU Kiel)
Exercises in a nice programming environment
Zurich: Programming environment Kara
Solve tasks by programming a little ladybird
Advantages: Attractive task, good identification

Programming with: Finite Automata, Java
Our approach: Erlang
Lessons

- basics: sequences, simple commands, case, recursion
- parameters (variables, integers, booleans)
- data structures (tuples, lists)
- modules, concurrency, distribution
- project: design and implementation of chat
Basic Language

\[
Prg ::= \text{Rule Prg} \\
| \text{Rule}
\]

\[
\text{Rule} ::= \text{Func (} \rightarrow \text{Cmds -}}
\]

\[
\text{Cmds} ::= \text{Cmd , Cmds} \\
| \text{Cmd}
\]

\[
\text{Cmd} ::= \text{go() | take() | mark() | nothing() | turn(Dir) | Func()} \\
| \text{case Dir() of} \\
| \quad Pat_1 \rightarrow e_1;
| \quad \ldots
| \quad Pat_n \rightarrow e_n \\
\text{end}
\]

\[
\text{Dir} ::= \text{left | right | front}
\]

\[
\text{Pat} ::= \text{free | shamrock | agaric | tree | border} \\
| \text{=}
\]

\[
\text{Func} ::= [a - z] [a - z, A - Z, 0 - 9]^*
\]
Exercises 1

[Diagram of two grids with shamrocks and a ladybug.]
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
Non-Tail Recursion

```
start() -> case front() of
    shamrock -> go(), take(),
                start(),
                mark(), go();
  _              -> go()
end.
```

Recursion is not difficult.

Recursion is cool!

Frank Huch (CAU Kiel)
Learning Erlang
15. Oktober 2007 11 / 18
Non-Tail Recursion

Recursion is cool!

Recursion is not difficult.

start() -> case front() of
    shamrock -> go(), take(),
              start(),
              mark(), go();
    _ -> go();
end.
Exercises 5
Excercises 5

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Frank Huch  (CAU Kiel)  Learning Erlang  15. Oktober 2007  12 / 18
Lessons

- basics: sequences, simple commands, case, recursion
- parameters (variables, integers, booleans)
- data structures (tuples, lists)
- modules, concurrency, distribution
- project: design and implementation of chat
Lessons

- basics: sequences, simple commands, case, recursion
- parameters (variables, integers, booleans)
- data structures (tuples, lists)
- modules, concurrency, distribution
- project: design and implementation of chat
Lessons

- basics: sequences, simple commands, case, recursion
- parameters (variables, integers, booleans)
- data structures (tuples, lists)
- modules, concurrency, distribution
- project: design and implementation of chat
Opportunities of Using Erlang in Our Approach

- only one programming language
- simple syntax
- no confusing types
- abstraction (reuse of function definitions)
- only recursion
- availability of Erlang

Problems:
- error messages
- old-fashioned GUI