Learning Programming with Erlang or Learning Erlang with Ladybirds

Frank Huch

Christian-Albrechts-University of Kiel

15. Oktober 2007

Learning Erlang with Ladybirds

Computer Science in Germany

- students in computer science in Germany (also Kiel): less than 20% girls
- computer science not taught in every school

Our approach for improvement:

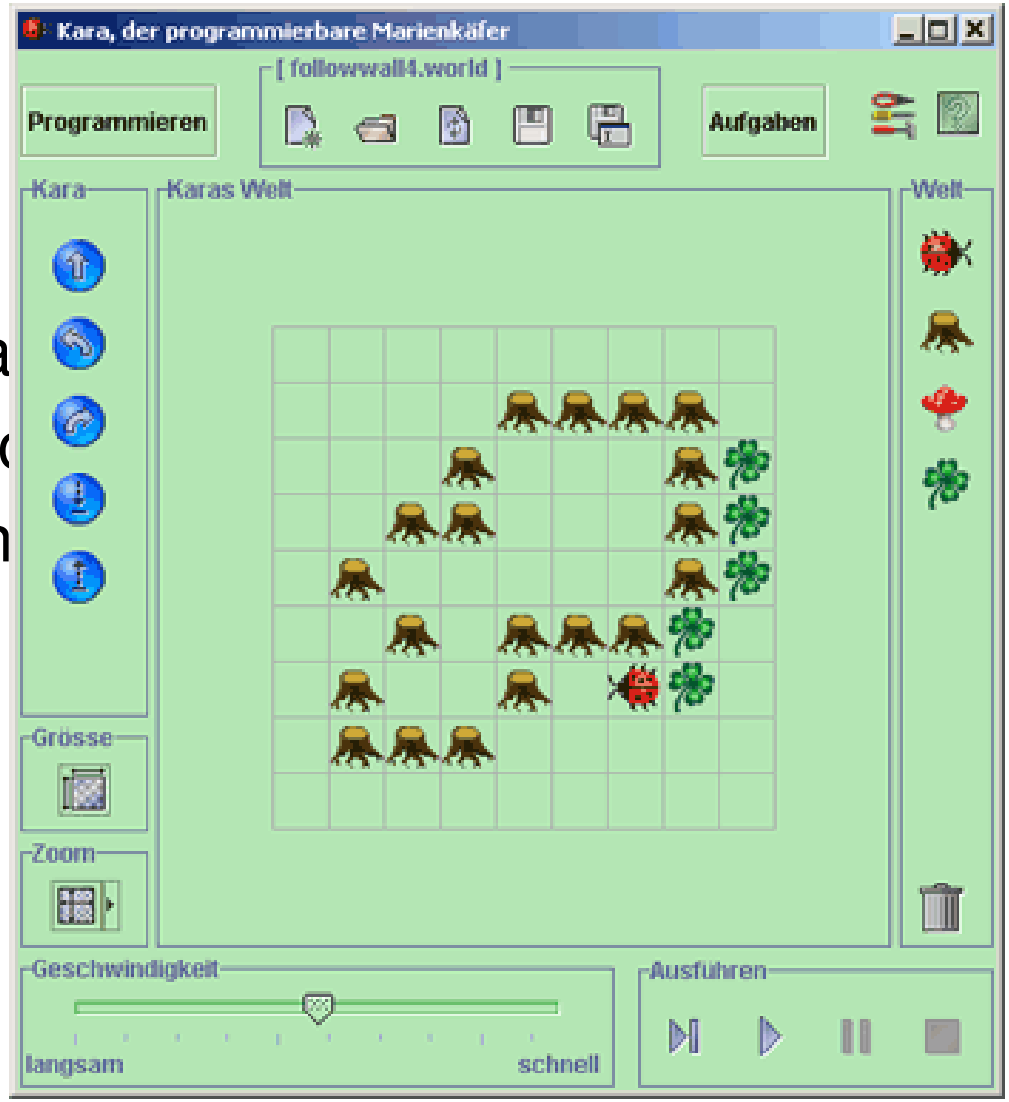
- One week course introducing computer science
- Talks of research groups, information about studying computer science, discussion with students and female computer scientist from industry, a trip to an IT company, and
- a programming course with a final project.

Teaching Programming

- Excercises in a nice programming environment
- Zurich: Programming environment Kara
- Solve tasks by programming a little ladybird

Teaching Programming

- Excercises in a nice progra
- Zurich: Programming envir
- Solve tasks by programmin

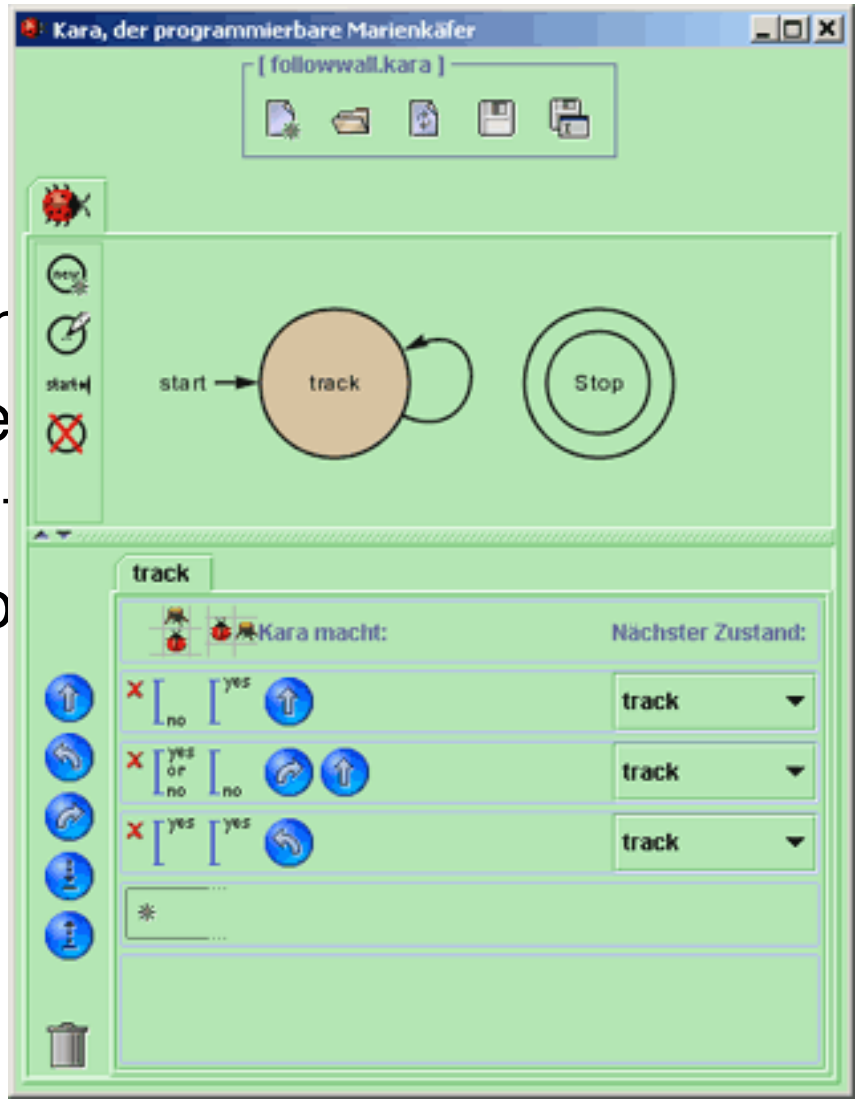


Teaching Programming

- Excercises 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 programmir
- Zurich: Programming environme
- Solve tasks by programming a li
- Advantages: Attractive task, goo



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

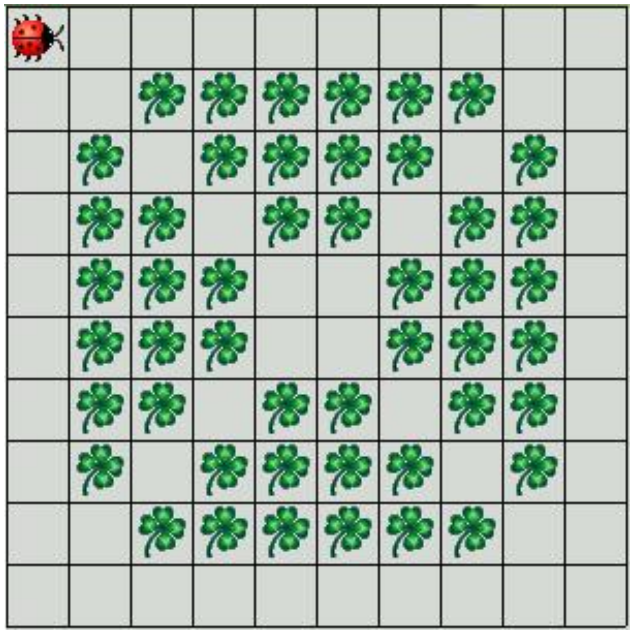
Lessons

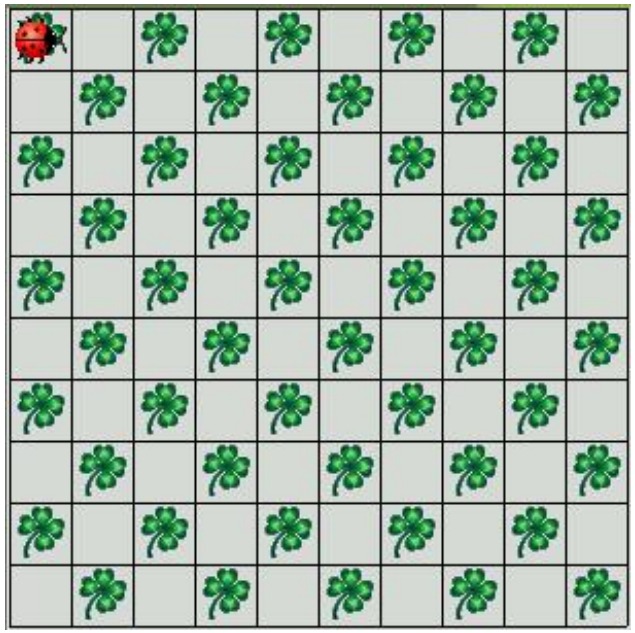
- basics: sequences, simple commands, case, recursion
- parameters (variables, integers, booleans)
- data structures (tuples, lists)
- modules, concurrency, distribution
- project: design and implementation of chat

Basic Language

Prg	::=	*Rule Prg*
	|	*Rule*
Rule	::=	*Func* `()` `->` *Cmds* `.`
Cmds	::=	*Cmd* `,` *Cmds*
	|	*Cmd*
Cmd	::=	`go()` | `take()` | `mark()` | `nothing()`
	|	`turn(`*Dir*`)` | *Func* `()`
	|	`case` *Dir* `()` `of`
		Pat_1 `->` e_1 `;`
		...
		Pat_n `->` e_n
		`end`
Dir	::=	`left` | `rigth` | `front`
Pat	::=	`free` | `shamrock` | `agaric` | `tree` | `border`
	|	`=`
Func	::=	$[\texttt{a}-\texttt{z}]\,[\texttt{a}-\texttt{z},\texttt{A}-\texttt{Z},\texttt{0}-\texttt{9}]^*$

Excercises 1

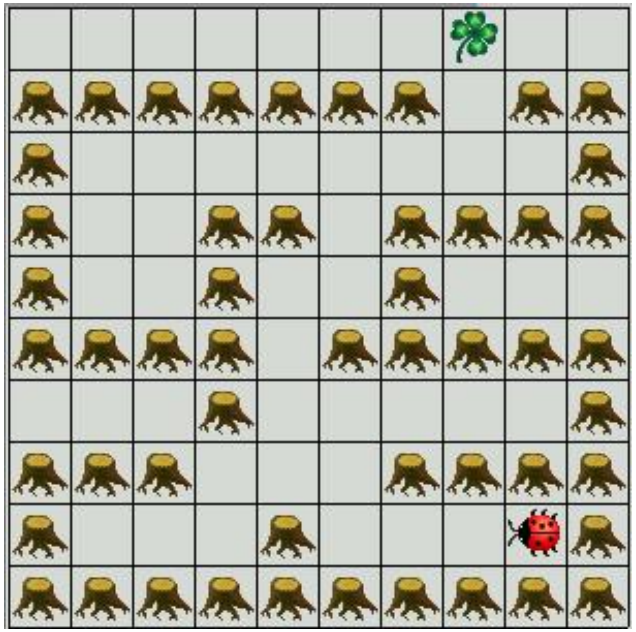




Excercises 2

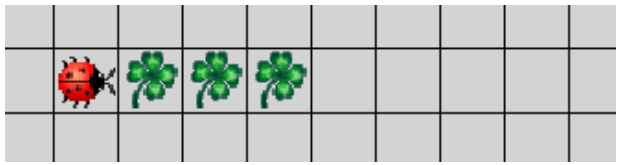
Excercises 3

Excercises 4



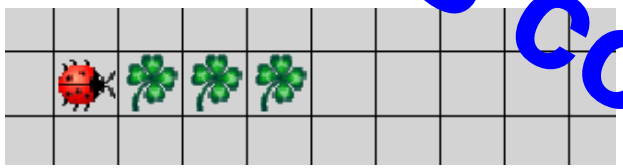
Non-Tail Recursion

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



Non-Tail Recursion

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



Recursion is not difficult.

Recursion is cool!

Excercises 5

Excercises 5

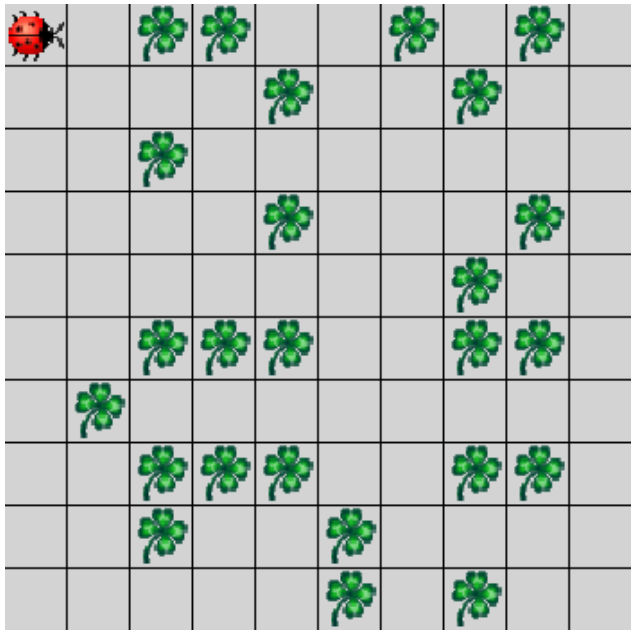
Excercises 5

Excercises 5

Lessons

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

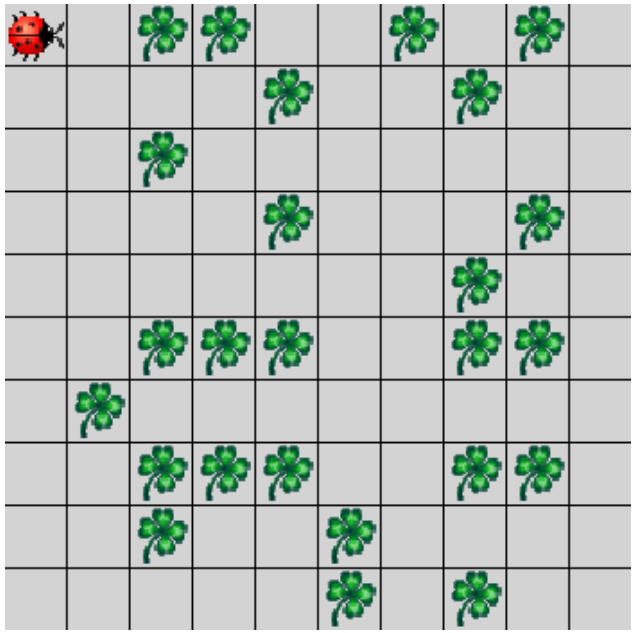
Excercises 7



Lessons

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

Excercises 7



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