Automatic Analysis of Industrial Robot Programs

OCaml Users and Developers Meeting 2012

Markus Weißmann,
Stefan Bedenk, Florian Pichlmeier,
Christian Buckl, Alois Knoll

Technische Universität München
Department of Informatics
Robotics and Embedded Systems
http://www6.in.tum.de

Audi

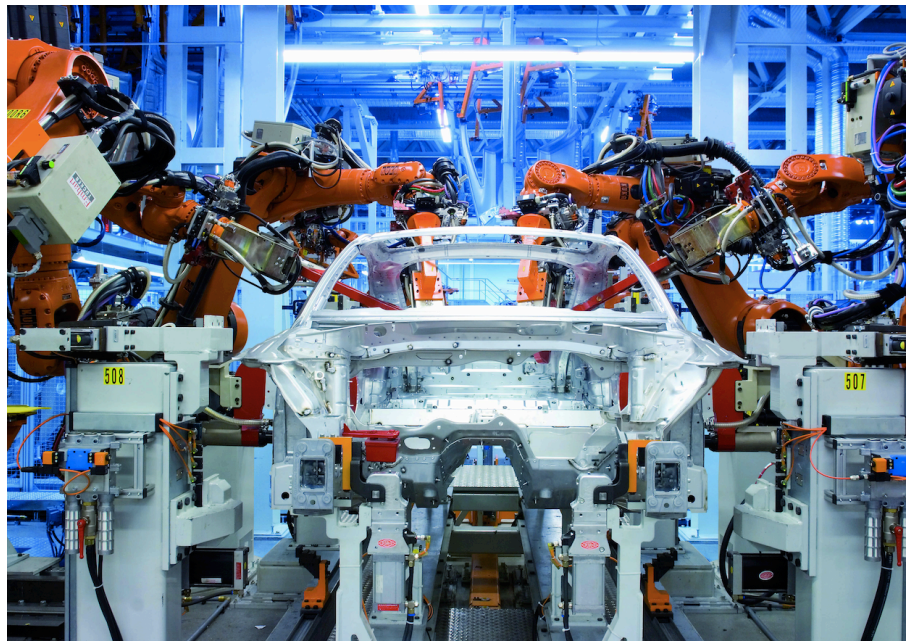
14.09.2012

Outline

Outline

Industrial Robot Systems – Car Body Production Lines

Audi



- Highly automated, complex systems
- Different car models on same production line
- Downtime extremely expensive
- Up to 12 robots working in a cell

Industrial Robot Systems – Topology

Distributed System of PLC and Robots

- Programming Logic Control (PLC) & robots work independently
- Incorrect handling of shared resources can lead to deadlocks and collisions (race conditions)

Robot Programming Language VKRC

Command	Semantics
variable = *expression*	Assignment of expression to variable
`GOTO LABEL` *id* = *expression*	Conditional jump to label *id*
`LABEL` *id*	Target of a jump
subroutine = *expression*	Conditional subroutine call
`REPEAT` *sub* = *n* `STOP` = *expr*	Call subroutine *n*-times
`WARTE BIS` *expression*	Wait until *expression* is met
n `PTP`	Move the robot arm to position *n*

Outline

Properties to verify

- Termination (deadlocks, ..)
- Freedom of collisions

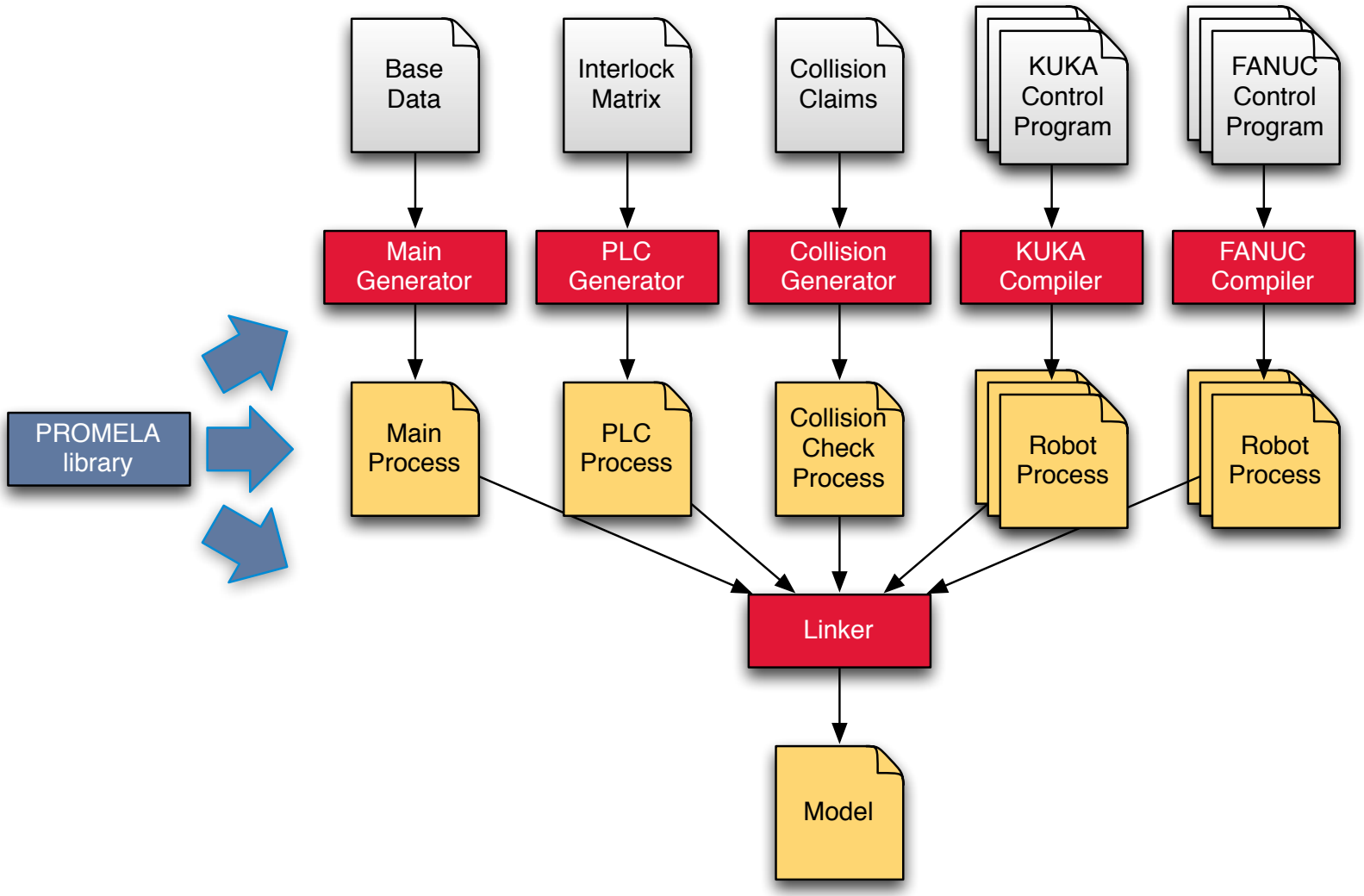
Robot 2

Robot 1	P1 -> P2	P2 -> P3	P3 -> P4
P1 -> P2	ok	⚡	⚡
P2 -> P3	ok	ok	⚡
P3 -> P4	ok	ok	ok

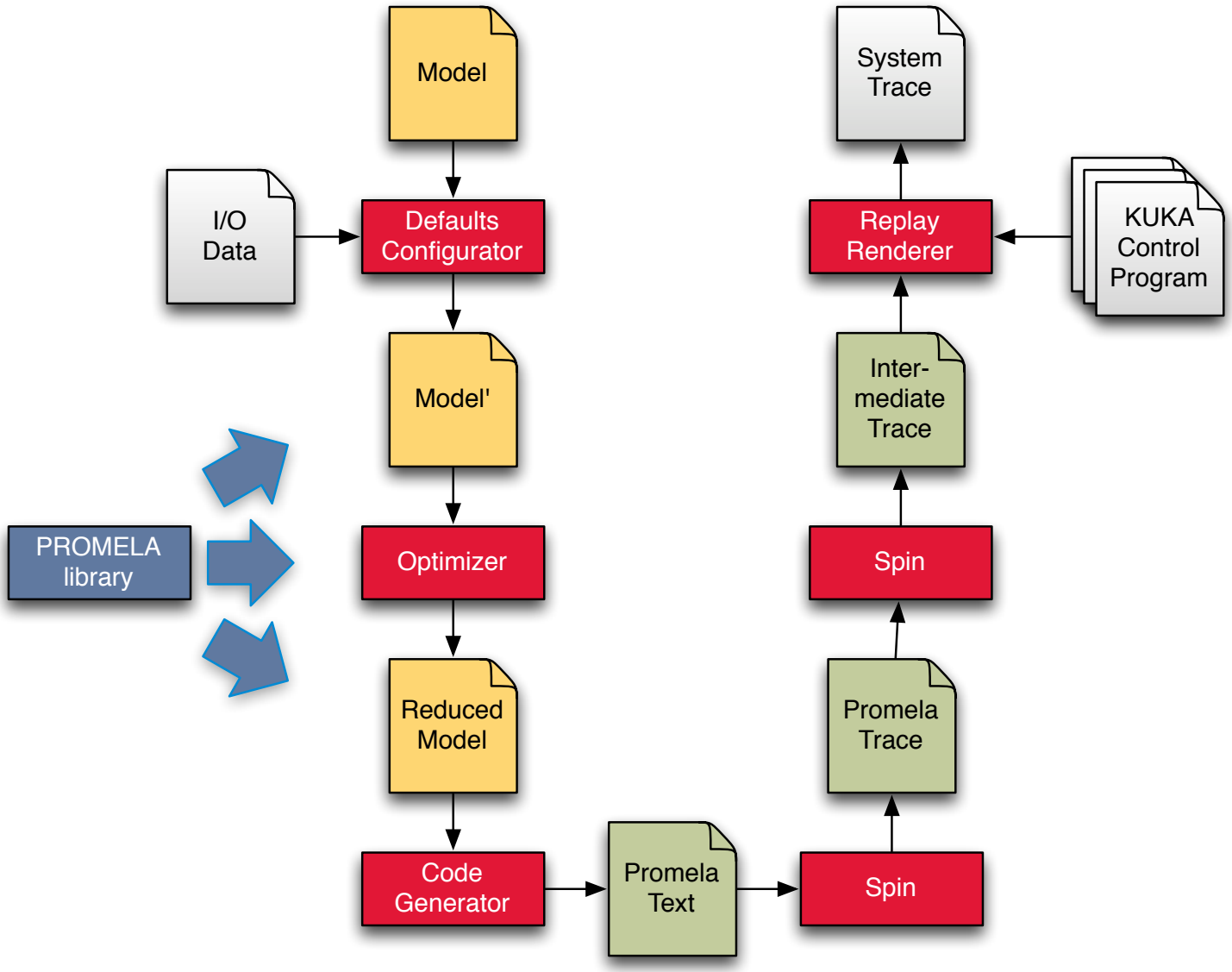
Claim for all potential collisions in Linear Temporal Logic

$$\square\neg((\text{target-position}_n = i) \wedge (\text{target-position}_m = k))$$

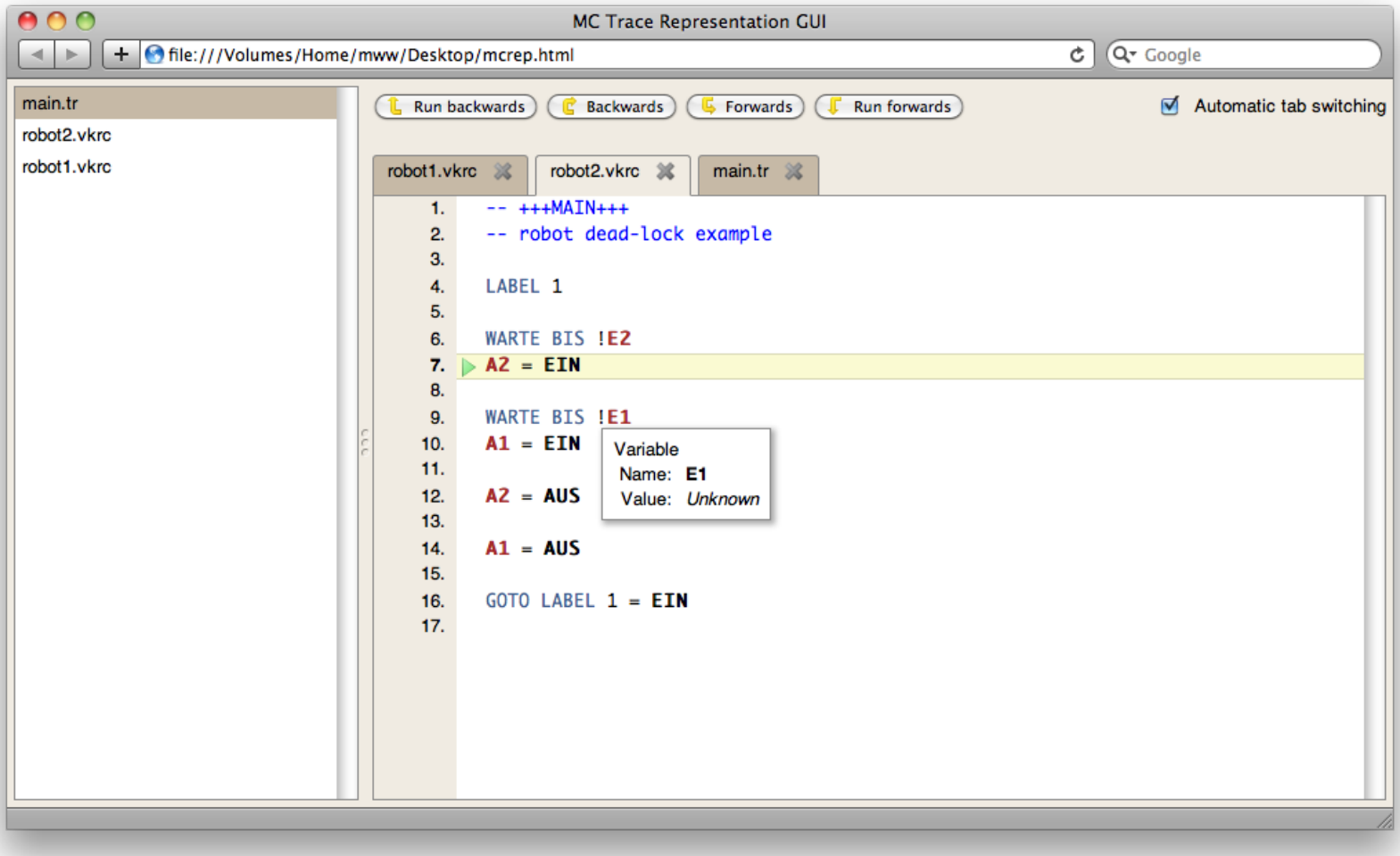
Model Generation and Extraction



Model Verification and Trace



Replay Renderer



Outline

Benchmarks

#r	#proc	state size	result	time	#states
4	5	112 bit	deadlock	1 sec	6.2×10^6
3	4	152 bit	non-termination	<1 sec	196
4	5	184 bit	ok	2 sec	1.5×10^6
4	5	196 bit	ok	6 sec	$3{,}5 \times 10^6$
4	5	202 bit	ok	50 sec	2.4×10^7
4	5	208 bit	deadlock	4 sec	10^6
9	11	678 bit	deadlock	<1 sec	5129
9	10	132 bit	assertion	<1 sec	32
10	11	506 bit	out-of-memory	~50 min	10^9

8-core Intel Xeon E5630 (2.53 GHz)
64 GByte of main memory

Locking Protocol – Normal Operation

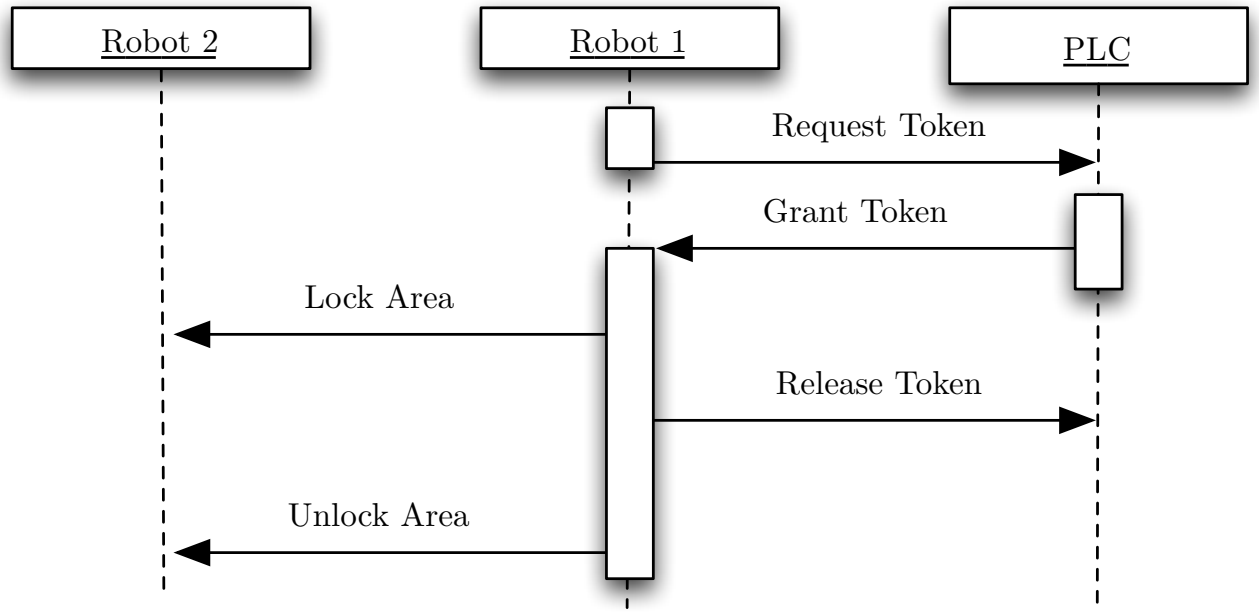


Figure 2: Robot 1 locks a resource for Robot 2

Locking Protocol – Potential Deadlock

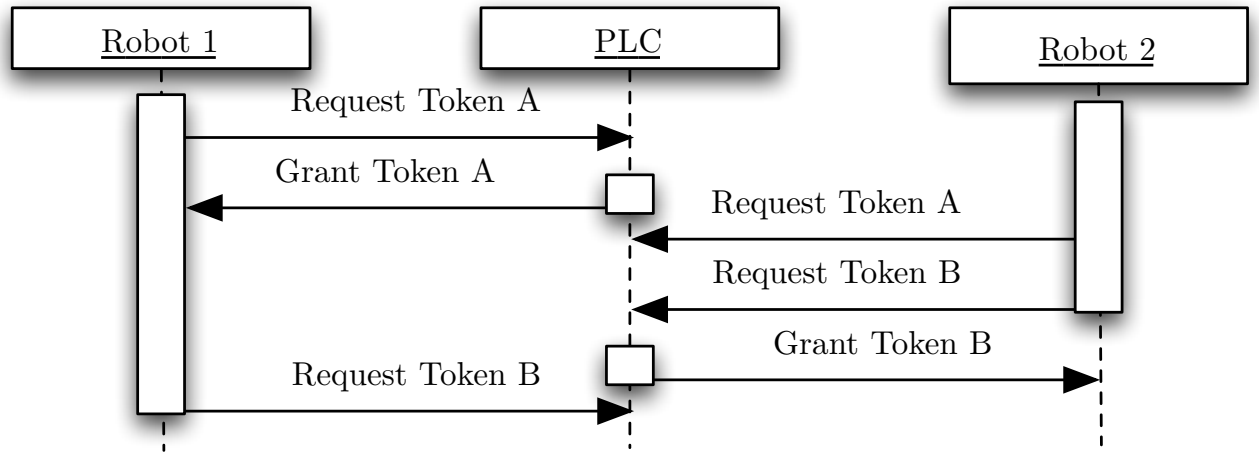


Figure 3: Robot can request second token without grant

Outline

OCaml Code

we use

- menhir
- ocamlgraph – control flow graph in compilers, code analysis
- ocaml-csv – vital to read Microsoft Excel exports
- bolt
- extlib
- qtest (from batteries) – for testing our own code
- oasis

and provide

- PROMELA library + tools (BSD3, OCamlForge)
- boolean expression simplifier (BSD3, OCamlForge)
- ocamlgraph algorithms: Fixpoint, Leaderlist, Contraction

PROMELA library example

```
(* Create one dining philosopher process *)
let dp_process ~max_id id =
  let module PE = Promela.Expression in
  let forkl, forkr = id, (id + 1) mod max_id in
  let check x =
    `Guard (PE.Binop (PE.Eq,
      (PE.Variable (fork_id x)), (PE.ConstInt 0))) in
  let assign v x =
    `Assign (fork_id x, None,(PE.ConstInt v)) in
  let take =  assign 1 in
  let release = assign 0 in
  let check_and_take x =
    `Atomic ((check x)::(take x)::[]) in
  Promela.Process.create ~active:true (pid "PHIL%d" id)
    [check_and_take forkl; check_and_take forkr;
      release forkl; release forkr]
```

ocamlgraph – fixpoint for data flow analysis

```
(* compute reachability on a directed graph *)
module Reachability = Graph.Fixpoint.Make(G)
(struct
  type vertex = G.E.vertex
  type edge = G.E.t
  type g = G.t
  type data = bool
  let direction = Graph.Fixpoint.Forward
  let equal = (=)
  let join = (||)
  let analyze _ = (fun x -> x)
end)

..
let vertex_is_reachable =
  Reachability.analyze is_root_vertex graph in
```

Outline

Experiences with OCaml

On the team

- strong OCaml community at TUM
- no OO in use
- try to write pure code
- refactoring is a breeze
- finding students for projects is hard

and with industrial partner

- Audi management feels strongly about results
- People in plant engineering are engineers, not computer scientists
- It is a research project

Outline

Conclusion

- Found multiple deadlock errors – even in protocol
- Formal Verification can even be cheaper than testing
- It is an option for analyzing large industrial robot systems
- Using OCaml for this project was a very good choice

Weißmann et al., Model Checking Industrial Robot Systems, SPIN 2011