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

Outline

1 Industrial Robot Systems
2 Formal Verification
3 Results
4 OCaml Code
5 Experiences with OCaml
6 Conclusion
Outline

1. Industrial Robot Systems
2. Formal Verification
3. Results
4. OCaml Code
5. Experiences with OCaml
6. Conclusion
Industrial Robot Systems – Car Body Production Lines

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>variable = expression</code></td>
<td>Assignment of expression to variable</td>
</tr>
<tr>
<td><code>GOTO LABEL id = expression</code></td>
<td>Conditional jump to label <code>id</code></td>
</tr>
<tr>
<td><code>LABEL id</code></td>
<td>Target of a jump</td>
</tr>
<tr>
<td><code>subroutine = expression</code></td>
<td>Conditional subroutine call</td>
</tr>
<tr>
<td><code>REPEAT sub = n STOP = expr</code></td>
<td>Call subroutine $n$-times</td>
</tr>
<tr>
<td><code>WARTE BIS expression</code></td>
<td>Wait until <code>expression</code> is met</td>
</tr>
<tr>
<td><code>n PTP</code></td>
<td>Move the robot arm to position $n$</td>
</tr>
</tbody>
</table>
Outline

1. Industrial Robot Systems
2. Formal Verification
3. Results
4. OCaml Code
5. Experiences with OCaml
6. Conclusion
Properties to verify

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

Claim for all potential collisions in Linear Temporal Logic

\[ \square \neg ((\text{target-position}_n = i) \land (\text{target-position}_m = k)) \]
Model Generation and Extraction

- Base Data
- Interlock Matrix
- Collision Claims
- KUKA Control Program
- FANUC Control Program
- Main Generator
- PLC Generator
- Collision Generator
- KUKA Compiler
- FANUC Compiler
- Main Process
- PLC Process
- Collision Check Process
- Robot Process
- Robot Process
- Linker
- Model

PROMELA library
Model Verification and Trace

- Model
  - Defaults Configurator
  - Model'
  - Optimizer
  - Reduced Model
  - Code Generator
  - Promela Text

- System Trace
  - Replay Renderer
  - Intermediate Trace
  - Spin
  - Promela Trace
  - Spin
  - KUKA Control Program

- I/O Data

- PROMELA library
Replay Renderer

Figure 1: Two robot programs in deadlock

Markus Weißmann, TUM
Outline

1. Industrial Robot Systems
2. Formal Verification
3. Results
4. OCaml Code
5. Experiences with OCaml
6. Conclusion
### Benchmarks

<table>
<thead>
<tr>
<th>#r</th>
<th>#proc</th>
<th>state size</th>
<th>result</th>
<th>time</th>
<th>#states</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>112 bit</td>
<td>deadlock</td>
<td>1 sec</td>
<td>$6.2 \times 10^6$</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>152 bit</td>
<td>non-termination</td>
<td>&lt;1 sec</td>
<td>196</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>184 bit</td>
<td>ok</td>
<td>2 sec</td>
<td>$1.5 \times 10^6$</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>196 bit</td>
<td>ok</td>
<td>6 sec</td>
<td>$3.5 \times 10^6$</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>202 bit</td>
<td>ok</td>
<td>50 sec</td>
<td>$2.4 \times 10^7$</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>208 bit</td>
<td>deadlock</td>
<td>4 sec</td>
<td>$10^6$</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>678 bit</td>
<td>deadlock</td>
<td>&lt;1 sec</td>
<td>5129</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>132 bit</td>
<td>assertion</td>
<td>&lt;1 sec</td>
<td>32</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>506 bit</td>
<td>out-of-memory</td>
<td>~50 min</td>
<td>$10^9$</td>
</tr>
</tbody>
</table>

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

1. Industrial Robot Systems
2. Formal Verification
3. Results
4. OCaml Code
5. Experiences with OCaml
6. Conclusion
OCaml Code

we use

- menhir
- ocamlgraph – control flow graph in compilers, code analysis
- ocaml-csv – vital to read Microsoft Excel exports
- bolt
- exlib
- 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

1. Industrial Robot Systems
2. Formal Verification
3. Results
4. OCaml Code
5. Experiences with OCaml
6. Conclusion
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

1. Industrial Robot Systems
2. Formal Verification
3. Results
4. OCaml Code
5. Experiences with OCaml
6. Conclusion
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