Formal Verification of Complex Robotic Systems on Resource-Constrained Platforms

Mohammed Foughali, Bernard Berthomieu, Silvano Dal Zilio, Pierre-Emmanuel Hladik, Félix Ingrand, Anthony Mallet
LAAS-CNRS/Université de Toulouse
Problem Definition

Autonomous robots -> perform as wanted?
Autonomous robots -> perform as wanted?
Problem Definition

Formal verification offers mathematical guarantees

X Non-understandable
X Time-consuming, error-prone formalization
X Scalability

Disconnection between the communities
Problem Definition: Robotic Software Layers

- Autonomous system software levels:
  - Decisional layer
    - Deals with high-level missions such as planning, acting, etc.
    - Often formal
  - Functional layer
    - Interacts directly with sensors and actuators
    - Deployed via non formal frameworks
    - Little has been done to formally verify its components
GenoM

- Services (control flow)
- Ports (data flow)
- Activities (fsm)
- Control task
- Execution tasks
Example 2: Quadcopter

- 5 components
- 6 ports
- 5 control task and 8 execution tasks
- >35 services
LAAS/RIS
- Functional level: GenoM
  - Modules
    - Services (control flow)
    - Ports (data flow)

LAAS/Vertics
- Fiacre/TINA framework for time-constrained distributed/concurrent systems

Fiacre

Functional Robotic Software

Formal Methods

Automatic Synthesis: GenoM -> Fiacre / TINA (Foughali et al. ICFEM 2016)
Automatic Synthesis: GenoM -> Fiacre/TINA
(Foughali et al. ICFEM 2016, Foughali ACSD 2017)

GenoM Models:
.idl & .gen

Fiacre Model of the Funct. Level

Fiacre/Model template
(GenoM/Pocolibs to Fiacre)

Fix the model

Real-time properties

TINA tools

Analysis

Fiacre Model

Fix the model

Fiacre/Model

template

(GenoM/Pocolibs

to Fiacre)

Real-time

properties

TINA tools

Analysis

Fiacre Model

Fix the model

Fiacre/Model

template

(GenoM/Pocolibs

to Fiacre)

Real-time

properties

TINA tools

Analysis
Problem definition: Hardware constraints

Wait.. how about the hardware?

Grrrrr why do you have to remind me?!
Problem definition: Hardware constraints

In the literature:

- **Schedulability analysis**
  - × Lack of automation
  - × Verification of other important behavioral/timed properties

- **Formal verification**
  - × Hardware constraints ignored
  - × Scalability

Develop a unified automated technique: integrate the hardware constraints into the Fiacre model
Particular Difficulty: Concurrency

Clients

Requests

Reports

Control Task

Execution Tasks

IDS

Ports

Requests

Reports

read/write
activity set_current_state() {
    task plan;
    codel<start> mv_current_state_read(in state, 
        out reference)
    wcet 1 ms
    yield ether;

    throw e_nostate;
};

activity launch {
    task exec;
    codel<start> mv_exec_start(out reference, out trajectory)
    yield wait;
    codel<wait> mv_exec_wait(in trajectory, in reference, out desired)
    yield pause::wait, wait, path, servo;

    codel<path> mv_exec_path(inout trajectory, 
        in reference, out desired, inout log)
    yield pause::path, wait;
    codel<servo> mv_exec_servo(inout reference, out desired, inout log)
    yield pause::wait;

    codel<stop> mv_exec_stop() yield ether;
};
Scheduling: enough cores (implicit)

process Timer_n (&tick_n: bool) is
  states start
  from start
  wait [PERIOD,PERIOD];
  tick_task_n := true;
  to start

process Taskmanager_n (... ,&tick_n: bool, &lock_n: bool) is
  states start, manage
  from start
  wait [0,0];
  on tick_n;
  tick_n := false;
  lock_n := false; /* pass the control to activities */
  to start

from manage
  wait [0,0];
  on lock_n; /* wait for activities to finish */
  to start

For task t, component m:
property sched_t_m is always ((m/t_manager/state manage) ⇒ not (m/t_manager/value tick_t))
Scheduling: FCFS

process scheduler (&fifo: queue N of 1..N, &launch: array 1..N of bool, &cores: 0..P) is

states start
from start
wait [0,0];
on (not empty fifo) and (cores > 0);
cores:= cores-1;
launch [first fifo]:= true;
 fifo := dequeue fifo;
to start

process Timer_n (&tick_n: bool) is

states start
from start
wait [PERIOD,PERIOD]; tick_task_n := true;
to start

process Taskmanager_n (...,&tick_n: bool, &fifo: queue N of 1..N, &launch: array 1..N of bool, &cores: 0..P) is

states ask, start, manage
from start
wait [0,0];
on tick_n; tick_n:= false;
to start

from ask
on launch[n]; wait [0,0];
lock_n:= false;
to manage

from manage
on lock_n; wait [0,0];
cores:= cores+1; launch[n]:= false

to start
function insert_sjf (q: queue N of 1..N, t: 1..N) :
queue N of 1..N is
var temp: 1..N
begin
    if (empty(q) or eet(t) < eet(first(q))) then
        return append(q,t)
    end if;
    temp:= first(q);
    return append(insert_sjf (dequeue(q), t), temp)
end

function eet
(t: 1.."[expr [llength [$c tasks]] + 1]">) : nat is
begin
    case t of
        1 →return 0 <'set k 2
    foreach task [$c tasks] {
        if {![catch {$task period}]} {'
            l <"$k"> →return <"[$task period]"/>
        } else {'
            l <"$k"> →return 0
        }
        incr k}'
    end
end
Results

For task t, module m:

property sched_t_m is always (not (m/t_manager/state start) \implies not (m/t_manager/value tick_t))

- Invariant: use the reduction by inclusion (faster)

* Hardware: Odroid-C0 board (4 cores)
All periodic tasks schedulable on the given hardware (both schedulers)

- Less cores?

**FCFS**

<table>
<thead>
<tr>
<th>Cores</th>
<th>SCG (size)</th>
<th>SCG (time)</th>
<th>sched_main (MikroKopter)</th>
<th>sched_main (NHFC)</th>
<th>sched_publish (OptiTrack)</th>
<th>sched_io (POM)</th>
<th>sched_filter (POM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>7338151</td>
<td>351.840s</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>3</td>
<td>10459826</td>
<td>485.764s</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>2</td>
<td>10788413</td>
<td>391.040s</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>1</td>
<td>40545</td>
<td>0.880s</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

**SJF**

<table>
<thead>
<tr>
<th>Cores</th>
<th>SCG (size)</th>
<th>SCG (time)</th>
<th>sched_main (MikroKopter)</th>
<th>sched_main (NHFC)</th>
<th>sched_publish (OptiTrack)</th>
<th>sched_io (POM)</th>
<th>sched_filter (POM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6210003</td>
<td>301.691s</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>3</td>
<td>7986495</td>
<td>333.289s</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>2</td>
<td>7008957</td>
<td>244.609s</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>1</td>
<td>32049</td>
<td>0.670s</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
✓ Possibility to verify other important real-time properties on the same model (as in Foughali et al. 2016)

Examples:
- Responsiveness of control task and aperiodic tasks
- Bounded ports update
Conclusion

✓ Summary:

✓ Schedulability verified automatically

✓ Important real-time properties verified on the actual hardware

✓ Unified environment for automated verification considering the real hardware-software setting
Future Work

Future work:

- Investigate optimized cooperative schedulers (HRRN, EDF, etc.)
- Extend the UPPAAL template
- Use the new models with Hippo for enforcement of properties
Part of this work is funded by the H2020 European project CPSE Labs under grant agreement No 644400

Thanks for your attention

questions are ¬ (¬ welcome)

–Mohammed