4 Years of Bridging the Gap

Formal Methods, Industry, and Distributed Systems

Robert Rubbens (but, call me Bob!)

2025-09-05

The topics

- About me
- About my research (so far)
- About what's next

About me

About me

Personal interests:

- Things related to computers, math, logic
- Piano, DnD
- Bouldering, Running
- ... Self-hosting stuff?

Please don't pwn me 😊

Availability:

- Mon: off
- Tue/wed: at FMT (Zi3082)
- Thu/Fri: at SCS (Zi2042)

Research interests

- Computer science!
- Gamedev and cybersecurity pre-bachelor
- Programming language design & theory after module 8

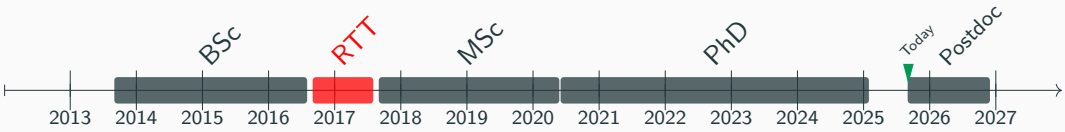
These days, anything related to programming languages and logic:

- Program verification, SMT, SAT
- Type systems, PL semantics
- Dependent type theory, interactive theorem proving

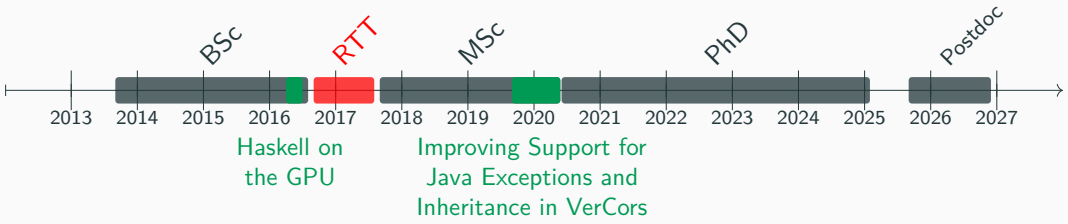
Happy to rekindle the cybersecurity flame :)

About my research (so far)

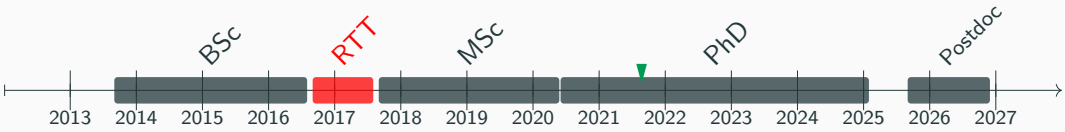
Timeline


BSc
RTT
MSc
PhD
Today
Postdoc
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027

Timeline

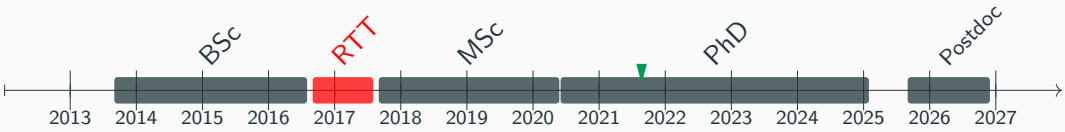

BSc
RTT
MSc
PhD
Postdoc
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
Haskell on the GPU
Improving Support for Java Exceptions and Inheritance in VerCors

Timeline



"Modular Transformation of Java Exceptions Modulo Errors"

Timeline

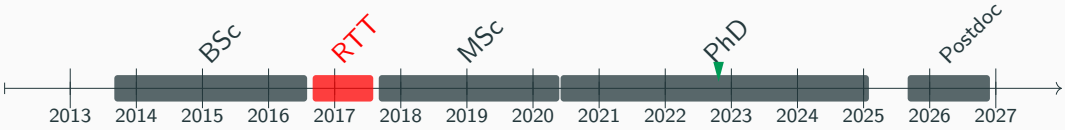


"Modular Transformation of Java Exceptions Modulo Errors"

```
L: while (c) {
   ... break L; ...
}
```

```
try { while (c) {
   ... throw new L(); ...
} } catch (L e) { }
```

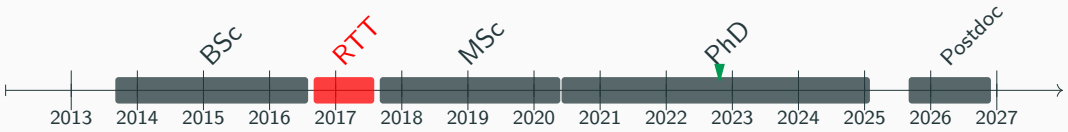
Timeline



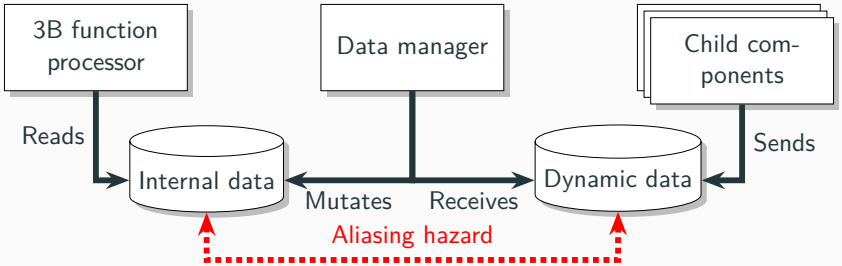
"On Deductive Verification of an Industrial Concurrent Software Component with VerCors" (invited)

- Tunnel software for "Blankenburgverbinding"
- Implemented according to the "BSTTI"

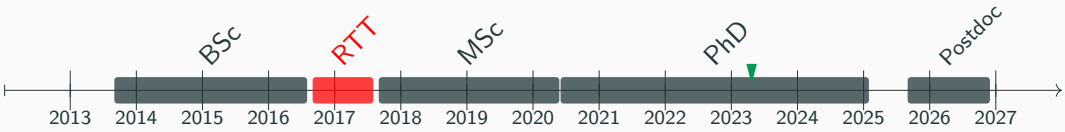
Timeline



"On Deductive Verification of an Industrial Concurrent Software Component with VerCors" (invited)

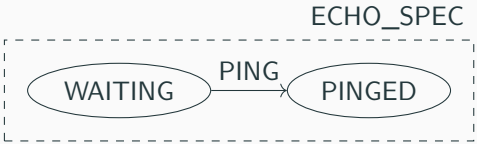


Timeline

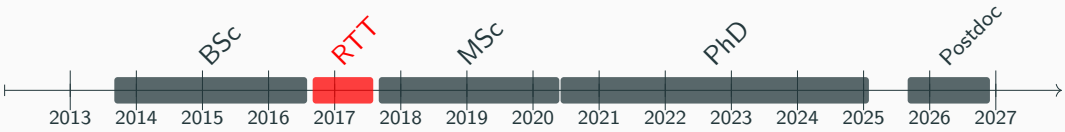


"JavaBIP meets VerCors: Towards the Safety of Concurrent Software Systems in Java"

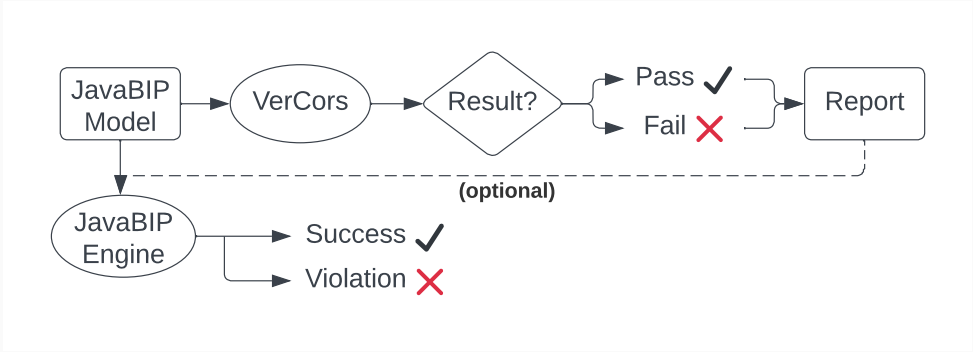
```
@Port(name = PING, type = PortType.enforceable)
@ComponentType(initial = WAITING, name = ECHO_SPEC)
public class Echo {
  @Transition(name = PING, source = WAITING, target = PINGED)
  public void ping () {
    System.out.println(this + ": pong");
  }
}
```



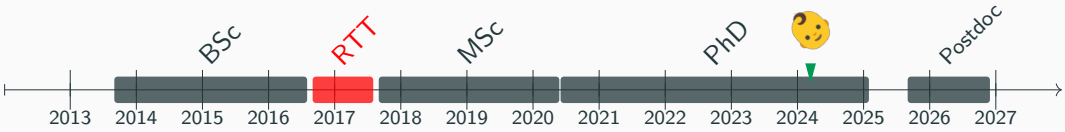
Timeline



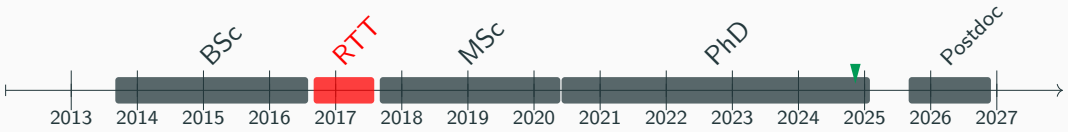
"JavaBIP meets VerCors: Towards the Safety of Concurrent Software Systems in Java"



Timeline


BSc
RTT
MSc
PhD
Postdoc
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027

Timeline



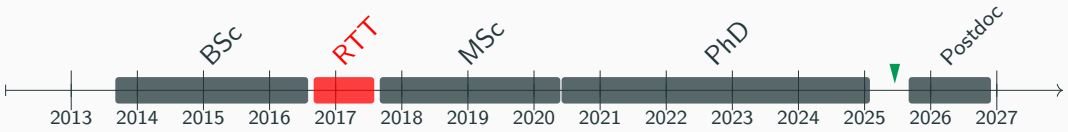
"VeyMont: Choreography-Based Generation of Correct Concurrent Programs with Shared Memory"

```
choreography incrStore(Store s) {
  endpoint a = Role(s);
  endpoint b = Role(null);
  requires a.s.x > 0;
  ensures a.s.x > 2;
  run {
    a.s.x := a.s.x + 1;
    communicate a.s -> b.s;
    b.s.x := b.s.x + 1;
    communicate b.s -> a.s;
  }
}
```

```
void incrStore_a(Role a, Role b, Channel chan) {
  run {
    a.s.x := a.s.x + 1;
    chan.send(a.s);
    a.s = chan.receive();
  }
}
```

```
void incrStore_b(Role a, Role b, Channel chan) {
  run {
    b.s = chan.receive();
    b.s.x := b.s.x + 1;
    chan.send(b.s);
  }
}
```

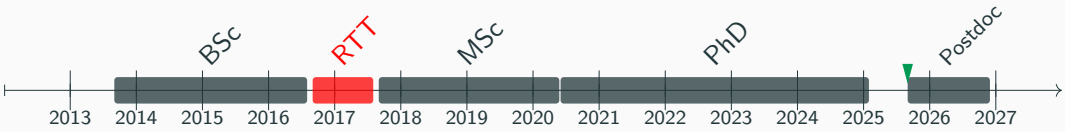
Timeline



"Verified Parameterized Choreographies"

```
choreography incrStoreN(Store s, N) {
  endpoint a[tid := 0 .. N] = Role(s, tid);
  endpoint b[tid := 0 .. N] = Role(null, tid);
  requires a.s.x > 0;
  ensures a.s.x > 2;
  run {
    a.s.x := a.s.x + 1;
    communicate a[i: 0 .. N].s -> b[i].s;
    b.s.x := b.s.x + 1;
    communicate b[i: 0 .. N].s -> a[i].s;
  }
}
```

Timeline



- SDLC, security properties (from CAPEC and ATT&CK), and check with formal methods (Event-B)
- Determine and/or quantify if a vulnerability is “dormant” or “live”

Things I look forward to:

1. More lenient context (false positives/negatives, or design level)
2. Easier to explain to family :-)