Markus A. Kuppe

Engineer TLA+ Project
RiSE group

March 08, 2019

"Write down a spec." (not just a hint but a principle)

Butler Lampson

TLA$^+$ 10,000ft Above

- High-level specification language independent of any programming language
  - Design above the code level
- Isolated from any framework
  - Abstract away irrelevant details
- Universally applicable
  - Scales from function-level to concurrent and distributed systems
- Highly expressive because based on basic mathematics
  - Rapid progress
- PlusCal has more scaffolding if desired
  - Learners choose PlusCal

TLA$^+$ 10,000ft Above

- High-level specification language independent of any programming language
  - Design above the code level
- Isolated from any framework
  - Abstract away irrelevant details
- Universally applicable
  - Scales from function-level to concurrent and distributed systems
- Highly expressive because based on basic mathematics
  - Rapid progress
- PlusCal has more scaffolding if desired
  - Learners choose PlusCal

TLA$^+$ 10,000ft Above

- High-level specification language independent of any programming language
  - Design above the code level
- Isolated from any framework
  - Abstract away irrelevant details
- Universally applicable
  - Scales from function-level to concurrent and distributed systems
- Highly expressive because based on basic mathematics
  - Rapid progress
- PlusCal has more scaffolding if desired
  - Learners choose PlusCal

TLA$^+$ 10,000ft Above

- High-level specification language independent of any programming language
  - Design above the code level
- Isolated from any framework
  - Abstract away irrelevant details
- Universally applicable
  - Scales from function-level to concurrent and distributed systems
- Highly expressive because based on basic mathematics
  - Rapid progress
- PlusCal has more scaffolding if desired
  - Learners choose PlusCal

TLA$^+$ 10,000ft Above

- High-level specification language independent of any programming language
  - Design above the code level
- Isolated from any framework
  - Abstract away irrelevant details
- Universally applicable
  - Scales from function-level to concurrent and distributed systems
- Highly expressive because based on basic mathematics
  - Rapid progress
- PlusCal has more scaffolding if desired
  - Learners choose PlusCal

Catching Bugs Before They Are Implemented

- First release of the Xbox 360
- MSR summer intern spec'ed IBM's memory coherence protocol
- *Writing* the spec revealed a subtle bug
- IBM acknowledge the bug only after several weeks
- Chips would have deadlocked after $\sim$ 4 hours of use
- Xbox Christmas launch would have been missed

Catching Bugs Early

- First release of the Xbox 360
- MSR summer intern spec'ed IBM's memory coherence protocol
- *Writing* the spec revealed a subtle bug
- IBM acknowledge the bug only after several weeks
- Chips would have deadlocked after $\sim$ 4 hours of use
- Xbox Christmas launch would have been missed

Finding Bugs Above The Code Level

Cheng Huang, Principle Software Engineer Manager:

> *"[...] We had a lock-free data structure implementation which was carefully design & implemented, went through thorough code review, and was tested under stress for many days. As a result, we had high confidence about the implementation. We eventually decided to write a TLA+ spec, not to verify correctness, but to allow team members to learn and practice PlusCal. So, when the model checker reported a safety violation, it really caught us by surprise."*

- TLA+ complements *program* verification such as F*.

Finding Bugs Above The Code Level

Cheng Huang, Principle Software Engineer Manager:

> *"[...] We had a lock-free data structure implementation which was carefully design & implemented, went through thorough code review, and was tested under stress for many days. As a result, we had high confidence about the implementation. We eventually decided to write a TLA+ spec, not to verify correctness, but to allow team members to learn and practice PlusCal. So, when the model checker reported a safety violation, it really caught us by surprise."*

- TLA+ complements *program* verification such as F*.

Finding Bugs Above The Code Level

Cheng Huang, Principle Software Engineer Manager:

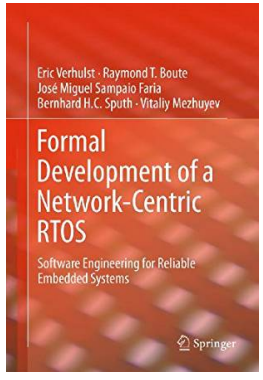
> *"[...] We had a lock-free data structure implementation which was carefully design & implemented, went through thorough code review, and was tested under stress for many days. As a result, we had high confidence about the implementation. We eventually decided to write a TLA+ spec, not to verify correctness, but to allow team members to learn and practice PlusCal. So, when the model checker reported a safety violation, it really caught us by surprise."*

- TLA+ complements *program* verification such as F*.

Cleaner Architecture

Verhulst [2011], Head OpenComRTOS development group:

> *"The [TLA+] abstraction helped a lot in coming to a much cleaner architecture (we witnessed first hand the brain washing done by years of C programming). One of the results was that the code size is about 10x less than [in the previous version]"*

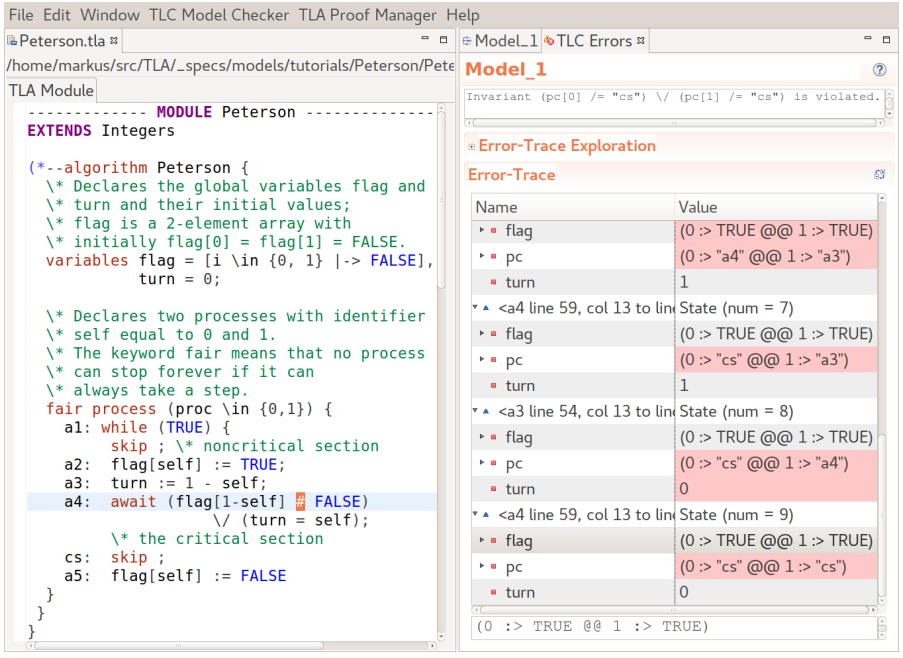


Adopting TLA+

- Low cost of adoption (no strategic bet)
  - Quick ROI even if lightly integrated into SDLC
- Adopters range from:
  - Single "TLA+ champions" (ad-hoc use)
  - Teams with dedicated verification engineers (integrated into SDLC)

Tooling

- Fully integrated ~~development~~ specification environment
- Push-Button Model Checking (TLC)
  - TLC answers question *if* a design is correct (finite model)
- Proof Assistant (TLAPS)
  - TLAPS verifies reasoning *why* a design is correct



Roadmap

- Ease-Of-Use
- Scale Model Checker to even larger Models
- Temporal Reasoning for Proof Assistant

- Externally:
  - Code Generation (PGo)
  - Z3-based Model Checker (Apalache)

"TLA+ is the most valuable thing that I've learned in my professional career. It has changed how I work [and] changed how I think [...]"

Chris Newcombe,
formerly Principal Engineer AWS

Bibliography I

Eric Verhulst. *Formal Development of a Network-Centric RTOS: Software Engineering for Reliable Embedded Systems*. Springer, New York, 2011. ISBN 978-1-4419-9735-7.