TLA+

Trust, but specify!

Markus Teufelberger

Graz Rust meetup 2018-07-05

TL-what?

TLA: Temporal Logic of Actions (~1980s)

TLA+: Specification language (~1993)

PlusCal: Algorithm language, “compiles”/translates to TLA+ (~2009)

TLC: Automated model checker for TLA+ specifications (~1999)

TLAPS (TLA+ proof system): Guided proof system for specifications written in TLA+ (~2008)

APALACHE: Abstraction-based parameterized TLA+ checker (~2015)

Temporal logic

Mixture of first-order and propositional logic with a few extra operators:

∧ ∨ ¬ TRUE FALSE ∀ ∃ etc.

[] P P is always TRUE (at every point in time)

<> P P is eventually TRUE (at one or many points in time)

<>[] P P will become and stay TRUE (it might be FALSE now)

P ~> Q P leads to Q (if P is TRUE, at some point later Q must be TRUE)

Temporal Logic of Arnold

Initial state:

ArnoldIsHere = FALSE
GetsToTheChoppa = TRUE

Which ones are true, according to Arnold?

A) <> ArnoldIsHere
B) [] ArnoldIsHere
C) <>[] ArnoldIsHere
D) GetsToTheChoppa ~> ArnoldIsHere

Guess the song

Initial state:

$\forall$ x $\in$ People: Call(x) = FALSE

me $\in$ People

Action:

<>Call(me)

TLA+

Syntax for TLA formulas

Mostly based in math notation, not programming languages
E.g. function definition: “==”, equality checking: “=”, inequality: “#”

Whitespace sensitive for logical junctions “\/” and “/\” (to save on (), you can indent subclauses)

Example:

https://gist.github.com/jparreira/76bb5d94de4822298a3dfddbf6bff950#file-wolf-sheep-cabbage-tla-full-tla

PlusCal

“Ugly pseudocode”

```
EXTENDS TLC

(* --algorithm hello_world
variable s \in {"Hello", "World!"};
Begin
  A:
    print s;
end algorithm; *)
```

In header of tla files, then converted to TLA+ below

`A:` is a label, everything within there happens at once.

TLC

Computing proofs and checking the provided specifications automatically

Complete evaluation - checking SHA3 in TLA+ will take a while

Closer to QuickCheck

TLAPS

Interactive proof assistant (TLC will have issues with checking something for all natural numbers, TLAPS might have a shot at it)

Integrates SMT/SAT solvers, but user has to guide proof - checking SHA3 in TLA+ with TLAPS will still take a while

Closer to Coq

APALACHE

“TLC is old, let’s do state of the art model checking”

Academic, TU Wien (Forsyte, Formal Methods in Systems Engineering)

Ressources

- https://lamport.azurewebsites.net/tla/tla.html (general overview)
- http://lamport.azurewebsites.net/video/videos.html (video series)
- https://medium.com/@jparreira/solving-the-wolf-sheep-and-cabbage-problem-using-tla-and-the-tlc-model-checker-28271a5afdfc (solve a logical puzzle)
- https://github.com/tlaplus (code, examples)
- https://learntla.com (guide)
- https://pron.github.io (in-depth blog posts, theory heavy)
- http://lamport.azurewebsites.net/tla/book.html (“Specifying Systems” - canonical source)
- http://forsyte.at/research/apalache/, https://github.com/konnov/apalache (APALACHE)