An Introduction to TLA+

Tom Gebert

November 17, 2017

What is a specification?

What is a specification?

- A specification is a document that describes the **behavior** or an application or system.
  - Says nothing about:
    - Performance
    - Implementation Details
    - Programming Language Choice
- Typically more mathematically oriented.
- Another layer of abstraction on top of your code

Why would you want a specification?

What is the appeal of a specification over a flow-chart and pseudo-code?

Why would you want a specification?

Flow-charts and pseudo-code are great tools, but they have some limitations:

- Relies a lot on "best guesses" and "gut feeings"
- Edge cases are easy to miss.
- There is no way to check to see if the design is correct.

Why would you want a specification?

In order to write and design systems with some guarantee of correctness, we should use the only time-proven tool for such things: Mathematics.

- Specification tools allow you to utilize the power of mathematics and apply it to the design
- By using mathematical notation, we create an inherently standardized way describing things

What is TLA+?

What is TLA+?

- TLA stands for the "Temporal Language of Actions"
- Designed by Leslie Lamport
  - Creator of Paxos and Logical Clocks
- A specification language and toolkit for describing sofware and testing algorithms.

What is TLA+?

TLA+ really describes three things:

- A language for specificying systems.
- A model checker for the systems that are specified.
- A rigorous proof system for checking specifications

What does a simple model look like?

Let's say we wanted to model a simple C program

```
int i = 0;
int main() {
    while(i < 10){
        i++;
    }
}
```

What does a simple model look like?

We fist define our program in terms of states.

- As the program begins, right before the i is defined, we can call that state “Begin”
- After i is assigned, we can call that state “Defined“
- After we’re done with the loop, can can call the final state “Done”

What does a simple model look like?

First we need to define our variables. Typically, you keep one variable to manage the program state we defined above, in addition to any needed for your algorithm. This variable is usually called “pc” for ”program controller”.

```
VARIABLES pc, i
```

What does a simple model look like?

We then need to define our initial state.

$$Init == pc = "Begin"$$

What does a simple model look like?

New states are designated with a ' symbol. We'll use this to "update" pc and "assign" i. We also use a disjunctive "AND" to add steps together.

```
Defined == /\ pc = "Begin"
           /\ i' = 0
           /\ pc' = "Initialize"
```

What does a simple model look like?

Now we are able to enter the loop. We check the state to ensure it's initialized, and then increment until we're not allowed to anymore.

```
Loop == /\ pc = "Initialize"
        /\ IF i < 10
              THEN /\ i' = i + 1
                   /\ pc' = "Initialize
              ELSE /\ pc' = "Done"
                   /\ i' = 10
```

What does a simple model look like?

Our model is done; Now we just need to tell TLA+ to effectively "loop" for us by defining our "Next" operation.

```
Next == Initialize \/ Loop
```

Demo!

Let me show you how this works in the TLA+ toolbox.

PlusCal

while it might be considered cheating, TLA+ has a system for programmers to more easily move over: PlusCal

PlusCal

To replicate what we had before, we could write something like:

```
i := 0;
while i < 10 do
i := i + 1;
end while
```

PlusCal

PlusCal gives you most of TLA+ and its mathematical beauty. It's cool. You should use it instead of pseudocode.