Rewriting TLA+ Fairness Conditions for Symbolic Model Checking

Sage Łuszczyk (k.luszczyk@student.tudelft.nl)

CSE3000 Research Project 2026 Q4 | Supervisors: Dr. Benedikt Ahrens, Dr. Anna Lukina

1 Liveness & Fairness

Example: Traffic Light

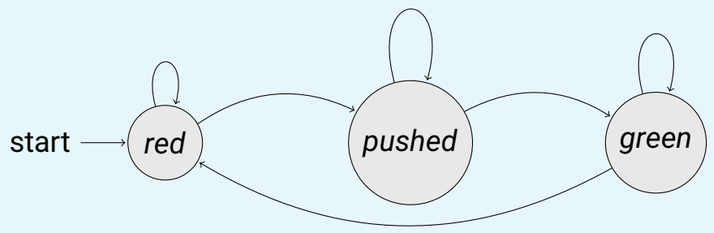

Figure: FSM representation of a traffic light, as modeled in TLA+

- When button pushed, can either switch to green or wait
- **Stuttering** – a transition where variables do not change
- **Property:** "It's always true that if the button is pushed, the traffic light eventually turns green"
- *EventuallyTurnsGreen* $\triangleq \Box(requestedGreen \Rightarrow \Diamond isGreen)$

- **TLA+** – language for describing systems and their expected properties, using discrete states
- **Liveness Property** – "something good happens eventually" (traffic light eventually turns green)
- **Weak Fairness** – assumption that if a non-stuttering transition ("action") is always possible, the system will eventually take it

$$WF_{vars}(Act) \triangleq \Diamond\Box(ENABLED\langle Act\rangle_{vars}) \Rightarrow \Box\Diamond\langle Act\rangle_{vars}$$

- **ENABLED** – predicate that checks if an action is possible (the light can only turn green if the button was pushed)

2 TLA+ Model Checker Limitations

- **TLC** – tool that checks if a system fulfills properties by exploring every possible state, which is expensive
- **Apalache** – similar, but instead builds a large logical formula based on how the system transitions between states, up to a certain point in the future
- **Problem:** Apalache does not support liveness properties that require fairness (traffic lights that can wait, but should turn green eventually), because it can not automatically handle ENABLED, must be specified directly

3 Research Questions

1. How can liveness properties with fairness conditions in TLA+ be rewritten to allow Apalache to check them?
   a. Can the given weak fairness condition be rewritten with the currently known method?
   b. If not, how can the method be modified to allow it?

4 Case Studies

1. Attempt to manually rewrite fairness condition
2. Automatically verify if formulas are equivalent with TLC
3. In case of failure, propose reason and rule modification

5 Results & Discussion

5.1 Traffic Light

```
Action ==
/\  /\  ~isGreen
    /\  requested
/\  /\  isGreen' = TRUE
    /\  requested' = FALSE
```

```
ENABLED ==
/\ /\  ~isGreen
   /\  requested
/\  vars # <<TRUE, FALSE>>
```

Figure: Relevant action with rewrite, which combines guard conditions with updating variables (primed variables are the value after transition)

```
Action ==
/\ Guard
/\ Assignment
```

```
ENABLED ==
/\ Guard
/\ NonStuttering
```

Figure: Simplified form of relevant action and its WF ENABLED rewrite

5.2 EWD840 Termination Detection

```
Action ==
\/ (Guard1 /\ Assignment1)
\/ (Guard2 /\ Assignment2)
```

```
ENABLED ==
\/ (Guard1 /\ NonStuttering1)
\/ (Guard2 /\ NonStuttering2)
```

Figure: Simplified form of EWD840's relevant action and its rewrite

5.3 Folklore Reliable Broadcast

```
Action ==
/\  Assignment1Guard
/\  \Exists Vars1 : Vars1Guard
/\  \/  DependentAssignment1
    \/  DependentAssignment2
```

```
ENABLED ==
/\  Assignment1Guard
/\  \Exists Vars1 :
    /\ Vars1Guard
    /\  \/ (DepAssign1Guard /\ NonStuttering1)
        \/ (DepAssign2Guard /\ NonStuttering2)
```

Figure: Simplified form of bcastFolklore's relevant action and its rewrite

6 Conclusions

- We expand the WF condition out, and rewrite the WF non-stuttering ENABLED predicate manually
- The literature method works for the Traffic Light case
- EWD840 contains independent sub-actions, which non-stuttering conditions should be distributed over
- bcastFolklore contains dependent assignments, which should be indented more than the initial assignment
- TLC model-checked ENABLED and WF as equivalent to built-in version, Apalache was able to verify liveness
- Proposed general method for future research: add indenting for every level of dependency, distribute dependent expressions across disjunctions

7 Limitations

- Improved literature method was based on small selection of case studies, thus is likely not fully general
- Proposed general method was synthesized from results and not itself verified, future work should apply it
- Metric of rewrite correctness was behavioral equivalence under TLC model checking, not strict logical equivalence
- Case studies used were simple, further complexity exists
- Focus was on specifications compatible with TLC