TLA+ Specification and Model Checking of the Agoric Smart Contracts Kernel

Andrey Kuprianov & Daniel Tisdall
Informal Systems
Vienna, Austria & London, UK
Email: {andrey, daniel}@informal.systems

Abstract

We present an industrial case study on specification and model checking of the Agoric smart contracts platform kernel. The platform is similar to a minimalistic operating system, where smart contracts live in user-space processes, called VATs, while the kernel is isolated from VATs, which make requests to the kernel via a syscall interface. In the talk we are going to present our experience of specifying the platform kernel in TLA+, our initial attempts on enhancing the specification with typing information, as well as the abstraction and refinement efforts needed to efficiently check interesting model properties.

1. Overview

Agoric [1] is considered to be among the pioneers and inventors of smart contracts. To ease the entrance into the smart contract world for developers, the Agoric platform allows to implement smart contracts in JavaScript, the most widely used programming language. In order to provide security guarantees the platform applies a unique architecture designed to eliminate the possibility of a multitude of typical smart contract bugs:

- The platform is written in Endo [2], a secure subset of JavaScript;
- The Swingset platform [3] is similar to a minimalistic operating system, in which the smart contracts live in user-space processes, called VATs, while the platform kernel [4] is isolated from VATs, which make requests to the kernel via a syscall interface;
- The kernel keeps track of user space objects and promises via a set of kernel tables, and enforces the system adherence to the OCAP (Object Capability) model properties.

Informal Systems [5] has been engaged by Agoric to specify the platform kernel in TLA+, and to verify that the kernel satisfies a set of requirements:

- OCAP model integrity (e.g. the “connectivity begets connectivity” property);
- Soundness of the kernel garbage collection protocol;
- Soundness of the kernel scheduling and metering protocols.

The kernel specification turned out to be highly nontrivial, spanning more than a thousand lines of dense TLA+, with more than 15 state variables required to account for various components of the kernel state, for the kernel execution and scheduling, as well as for the error handling. During the model creation, we have found the recent addition of type checking [6] into the Apalache model checker [7] to be of tremendous usefulness, helping to catch modelling errors early. On the other hand, while even the sole exercise of model construction was very useful for understanding the kernel, the model became too heavy to be model checked. Therefore, we employ abstractions, and construct abstract model variants for the purpose of verifying the above requirements, with refinement relations [8] connecting the abstract and precise model variants.

2. References

[1] The Agoric company: https://agoric.com

[2] The Endo secure Javascript platform: https://github.com/endojs/endo

[3] The Swingset smart contracts platform:
https://github.com/Agoric/agoric-sdk/tree/master/packages/SwingSet

[4] The Swingset kernel:
https://github.com/Agoric/agoric-sdk/tree/master/packages/SwingSet/src/kernel

[5] The Informal Systems company: https://informal.systems

[6] Tutorial on the Snowcat type checker:
https://apalache.informal.systems/docs/tutorials/snowcat-tutorial.html

[7] The Apalache model checker: https://apalache.informal.systems

[8] Leslie Lamport. Hiding, Refinement, and Auxiliary Variables:
https://lamport.azurewebsites.net/tla/hiding-and-refinement.pdf