A Formal Specification of the Cardano Ledger

Deliverable SL-D5

Jared Corduan <jared.corduan@iohk.io>

Polina Vinogradova <polina.vinogradova@iohk.io>

Matthias Güdemann <matthias.gudemann@iohk.io>

Project: Shelley Ledger

Type: Deliverable
Due Date: 15th October 2019

Responsible team: Formal Methods Team
Editor: Jared Corduan, IOHK
Team Leader: Philipp Kant, IOHK

Version 1.0
8th October 2019

Dissemination Level		
PU	Public	√
CO	Confidential, only for company distribution	
DR	Draft, not for general circulation	

Contents

List of Figures

Change Log

Rev.	Date	Who	Team	What
1	2019/10/08	Jared Corduan, Polina Vinogradova and Matthias Güdemann	FM (IOHK)	Initial version (0.1).
2	2019/10/08	Kevin Hammond	FM (IOHK)	Added cover page.
3	2020/11/17	Jared Corduan	FM (IOHK)	Removed unused deliverable outline image, set version to 1.0, and changed the reward calculation so that η takes d into account.
4	2021/05/17	Jared Corduan	FM (IOHK)	Added Example Illustration.
5	2021/06/14	Jared Corduan	FM (IOHK)	Added an errata section, fixed a typo in leader check and in the LEDGERS rule index, and synced the reward calculation with the implementation
6	2021/06/17	Jared Corduan	FM (IOHK)	Allow the pool influence parameter a_0 to be zero, remove all mentions of deposit decay, sync varibale names with code.
7	2021/08/27	Jared Corduan	FM (IOHK)	Fixed definitions in the header-only validation properties. CHAIN transition did not need to use previous protocol parameters.
8	2021/10/08	Jared Corduan	FM (IOHK)	The function createRUpd should get the pool parameters from the go snapshot. The TICKN rule was missing from the dependency diagram.
9	2021/11/08	Jared Corduan	FM (IOHK)	Fixed typo in the description of variable length encodings.
10	2021/12/13	Jared Corduan	FM (IOHK)	Re-wrote the MIR transitions to be more compact.
11	2022/01/20	Jared Corduan	FM (IOHK)	Fixed error in counting new pools for deposits and an error in the POOLREAP rule.
12	2022/01/26	Jared Corduan	FM (IOHK)	Specify seed operation and seedToSlot.
13	2022/01/31	Jordan Millar, Jared Corduan	FM (IOHK)	Fixed prose regarding the hash used in the epoch nonce. Added an item to the errata regarding this same hash.
14	2022/08/25	Jared Corduan	FM (IOHK)	Specify the txid function in the Appendix. Warn about re-serializing.
15	2023/03/23	Jared Corduan	FM (IOHK)	Add an errata section about individualP deposit tracking.

A Formal Specification of the Cardano Ledger

Jared Corduan
jared.corduan@iohk.io

Polina Vinogradova
polina.vinogradova@iohk.io

Matthias Güdemann
matthias.gudemann@iohk.io

March 23, 2023

Abstract

This document provides a formal specification of the Cardano ledger for use in the upcoming Shelley implementation. It is intended to underpin a Haskell executable specification that will be the basis of the initial Shelley release, and represents a core design and quality assurance document. It will be used to define properties and tests, and to provide the basis for strong formal assurance using mathematical proof techniques. The document defines the rules for extending the ledger with transactions that will affect both UTxO and stake delegation. Key properties that have been identified include the preservation of balances, absence of double spend, stakepool registration, and reward splitting.

List of Contributors

Nicolás Arqueros, Dan Bornside, Nicholas Clarke, Duncan Coutts, Ruslan Dudin, Sebastien Guillemot, Kevin Hammond, Vincent Hanquez, Ru Horlick, Michael Hueschen, Christian Lindgren, Yun Lu, Philipp Kant, Jordan Millar, Jean-Christophe Mincke, Damian Nadales, Ashish Prajapati, Nicolas Di Prima, Andrew Westberg.

1 Introduction

This document is a formal specification of the functionality of the ledger on the blockchain. This includes the blockchain layer determining what is a valid block, but does not include any concurrency issues such as chain selection. The details of the background and the larger context for the design decisions formalized in this document are presented in [SL-D1].

In this document, we present the most important properties that any implementation of the ledger must have. Specifically, we model the following aspects of the functionality of the ledger on the blockchain:

Preservation of value Every coin in the system must be accounted for, and the total amount is unchanged by every transaction and epoch change. In particular, every coin is accounted for by exactly one of the following categories:

- Circulation (UTxO)
- Deposit pot
- Fee pot
- Reserves (monetary expansion)
- Rewards (account addresses)
- Treasury

Witnesses Authentication of parts of the transaction data by means of cryptographic entities (such as signatures and private keys) contained in these transactions.

Delegation Validity of delegation certificates, which delegate block-signing rights.

Stake Staking rights associated to an address.

Rewards Reward calculation and distribution.

Updates The update mechanism for Shelley protocol parameters and software.

While the blockchain protocol is a reactive system that is driven by the arrival of blocks causing updates to the ledger, the formal description is a collection of rules that compose a static description of what a *valid ledger* is. A valid ledger state can only be reached by applying a sequence of inference rules and any valid ledger state is reachable by applying some sequence of these rules. The specifics of the semantics we use to define and apply the rules we present in this document are explained in detail in [FM-TR-2018-01] (this document will really help the reader's understanding of the contents of this specification).

The structure of the rules that we give here is such that their application is deterministic. That is, given a specific initial state and relevant environmental constants, there is no ambiguity about which rule should be applied at any given time (i.e. which state transition is allowed to take place). This property ensures that the specification is totally precise — no choice is left to the implementor and any two implementations must behave the same when it comes to deciding whether a block is valid.

2 Notation

The transition system is explained in [FM-TR-2018-01].

Powerset Given a set X, $\mathbb{P}\ \mathsf{X}$ is the set of all the subsets of X.

Sequences Given a set X, X^* is the set of sequences having elements taken from X. The empty sequence is denoted by ϵ. Given a sequence Λ, $\Lambda; \mathsf{x}$ is the sequence that results from appending $\mathsf{x} \in \mathsf{X}$ to Λ.

Functions $A \to B$ denotes a **total function** from A to B. Given a function f we write $f\ a$ for the application of f to argument a.

Inverse Image Given a function $f : A \to B$ and $b \in B$, we write $f^{-1}\ b$ for the **inverse image** of f at b, which is defined by $\{a \mid \ fa = b\}$.

Maps and partial functions $A \mapsto B$ denotes a **partial function** from A to B, which can be seen as a map (dictionary) with keys in A and values in B. Given a map $m \in A \mapsto B$, notation $a \mapsto b \in m$ is equivalent to $m\ a = b$. The $\emptyset$ symbol is also used to represent the empty map as well.

Map Operations Figure 1 describes some non-standard map operations.

Relations A relation on $A \times B$ is a subset of $A \times B$. Both maps and functions can be thought of as relations. A function $f : A \to B$ is a relation consisting of pairs $(a, f(a))$ such that $a \in A$. A map $m : A \mapsto B$ is a relation consisting of pairs (a, b) such that $a \mapsto b \in m$. Given a relation R on $A \times B$, we define the inverse relation R^{-1} to be all pairs (b, a) such that $(a, b) \in R$. Similarly, given a function $f : A \to B$ we define the inverse relation f^{-1} to consist of all pairs $(f(a), a)$. Finally, given two relations $R \subseteq A \times B$ and $S \subseteq B \times C$, we define the compostion $R \circ S$ to be all pairs (a, c) such that $(a, b) \in R$ and $(b, c) \in S$ for some $b \in B$.

Option type An option type in type A is denoted as $A^? = A + \Diamond$. The A case corresponds to the case when there is a value of type A and the $\Diamond$ case corresponds to the case when there is no value.

:= We abuse the **:=** symbol here to mean propositional equality. In the style of semantics we use in this formal spec, definitional equality is not needed. It is meant to make the spec easier to read in the sense that each time we use it, we use a fresh variable as shorthand notation for an expression, e.g. we write

$$s := slot + \mathsf{StabilityWindow}$$

Then, in subsequent expressions, it is more convenient to write simply s. It is not meant to shadow variables, and if it does, there is likely a problem with the rules that must be addressed.

In Figure 1, we specify the notation that we use in the rest of the document.

$$
\begin{array}{rlr}
\mathit{set} \lhd \mathit{map} & = \{k \mapsto v \mid k \mapsto v \in \mathit{map},\ k \in \mathit{set}\} & \text{domain restriction} \\
\mathit{set} \not\lhd \mathit{map} & = \{k \mapsto v \mid k \mapsto v \in \mathit{map},\ k \notin \mathit{set}\} & \text{domain exclusion} \\
\mathit{map} \rhd \mathit{set} & = \{k \mapsto v \mid k \mapsto v \in \mathit{map},\ v \in \mathit{set}\} & \text{range restriction} \\
\mathit{map} \not\rhd \mathit{set} & = \{k \mapsto v \mid k \mapsto v \in \mathit{map},\ v \notin \mathit{set}\} & \text{range exclusion} \\
A \triangle B & = (A \setminus B) \cup (B \setminus A) & \text{symmetric difference} \\
M \underrightarrow{\cup} N & = (\operatorname{dom} N \not\lhd M) \cup N & \text{union override right} \\
M \underleftarrow{\cup} N & = M \cup (\operatorname{dom} M \not\lhd N) & \text{union override left} \\
M \cup_{+} N & = (M \triangle N) \cup \{k \mapsto v_1 + v_2 \mid k \mapsto v_1 \in M \land k \mapsto v_2 \in N\} & \text{union override plus} \\
& & \text{(for monoidal values)}
\end{array}
$$

Figure 1: Non-standard map operators