Determinism of ledger updates

Polina Vinogradova
IOG

James Chapman
IOG

Andre Knispel
IOG

Orestis Melkonian
IOG & University of Edinburgh, UK

Extended abstract

2012 ACM Subject Classification Theory of computation → Operational semantics

Keywords and phrases determinism, ledger model, theory of changes

Funding This work was supported by Input Output (iohk.io) through their funding of the Edinburgh Blockchain Technology Lab.

1 Introduction

In the context of blockchain transaction processing and smart contract execution, determinism is usually taken to mean something like "the ability to predict locally, before submitting a transaction, the on-chain result of processing that transaction and its scripts". This is an important aspect of ledger design because users care about being able to accurately predict fees they will be charged, rewards they will receive from staking, outcomes of smart contract executions, etc. before submitting transactions. The purpose of this work is to formalize this property of ledgers, and study the constraints under which it can be guaranteed, thereby providing analysis tools and design principles for building ledgers whose transaction processing outcomes can be accurately forecast.

Blockchain ledger and consensus design relies on the definition of the transaction processing function itself being entirely deterministic. The impossibility of predicting the exact on-chain state transactions will be applied to, however, is due to unpredictable network propagation of transactions, resulting in an arbitrary order in which they are processed as the source of non-determinism. Determinism can, therefore, be formulated entirely in the language of transaction application commutativity, but we retain the commonly used blockchain terminology in this work.

We present an abstract ledger model capturing the architectural core shared by most blockchain platforms: a ledger is a state transition system, with valid transactions (or blocks) as the only transitions. We then formalize the definition of determinism in terms of this abstract functional specification of ledger structure, and use mathematical tools for analyzing them in order is to establish a way to reason about conditions under which the transaction order has no effect on the resulting state or parts thereof.

A similar construction to our ledger definition is presented in [4]. However, the focus there is on the ordering of state updates to optimize the execution of a particular consensus algorithm. Another related study of commutativity of state transitions is [1].

2 The abstract ledger model and determinism

A ledger is defined by its state type State, transaction type Tx, initial state $\mathsf{initState} : \mathsf{State}$, and a state update function $\mathsf{update} : \mathsf{Tx} \to \mathsf{State} \to \mathsf{State} \uplus \bot$.

A transaction tx is called *valid* in state s whenever $\mathsf{update}\ tx\ s \neq \bot$. A valid ledger state is one that is the result of applying a sequence of transactions to the initial state, where each transaction is valid in the state it is applied to. We call such a sequence a *trace*.

Defining valid state in this way allows us to make precise the nature of the discrepancy between a valid local and a valid on-chain state that may prevent a user from accurately predicting the consequences of their transaction being applied : since both states are the result of applying a valid sequence of transactions, the only reason they may be distinct is that their traces differ (usually because some more recent transactions have not yet been applied to the local state).

We present two definitions of determinism and establish a relation between them. The first definition, which we call **order-determinism**, requires that any two permutations of a list of transactions must produce the same state when applied to a given valid state, unless one (or both) of them produces an $\perp$. A ledger where the State is the collection of all multisets of Tx, and $\mathsf{update}\ tx\ s := s \uplus \{tx\}$, is an example of an order-deterministic ledger.

The second definition, called **update-determinism**, requires that if two transactions produce the same state when applied to a valid state, they must also produce the same state when applied to any other valid state. An example of a ledger that is neither is one with $\mathsf{State} = \mathsf{Tx} = \mathbb{B}$, and an update function that flips the boolean in the state if the transaction and state booleans match.

3 Mathematical models of abstract ledgers

An important part of future and ongoing determinism research is the application of mathematical tools and constructs to characterize determinism in ledgers, such as category theory and group theory. We define a one-object category of all valid states plus $\perp$ where the maps are specified by the transactions, as well as two other related categories. We also define a category of all ledgers, together with all maps between them that preserve the initial state and update function. We also define a free, finitely generated monoid whose underlying set, for a given ledger, consists of all dependent pairs of a list of transactions and the corresponding states computed by the update function.

The *theory of changes* is a framework for defining differentiation of functions specifying updates to data structures [2]. We adjust this formalism to partial functions so that it can be employed in the context of ledgers and ledger updates. The notion of a type representing a change set is difficult to define while remaining agnostic of the specific data structure, however, we observe that every permissible set of ledger changes corresponds exactly to a sequence of valid transactions, and update is the function that applies those changes. We apply this idea to formalize the colloquial determinism definition, and present a proof that the definition of determinism we give in terms of change sets and derivatives is equivalent to order-determinism. We then also use this differentiation formalism to analyze update-determinism and relate it to order-determinism.

So far, our reasoning has been agnostic of the content and structure of ledger states. In many cases, specific parts of the ledger are of interest for the study of determinism, such as an account or smart contract state. We introduce the concept of threads to formalize what we mean by ledger parts, as well as define what it means for an individual thread to be deterministic. We show that all threads must be order-deterministic in an order-deterministic ledger.

4 Discussion

Assumptions and limitations We use the term "transaction" to refer to the ledger state transition type for the reason that we make the assumption that most users are concerned with application of transactions to the ledger state. However, in practice, an atomic update of a blockchain's state is a block, which updates ledger data that transaction application cannot modify, such as the hash of the previous block or the current slot number. The notion of threads lets us formalize the relation between block-based and transaction-based updates, but we leave this for future work.

Our model is constructed to reflect the simplifying assumption that there is exactly one way to interface with a ledger — by applying a transaction, which is indeed the case in most normal circumstances. Our model is also strictly functional, so that the update function itself is necessarily deterministic. We also make the assumption that the update function does not evolve in any way.

Applications to Cardano. We can apply the tools for ledger analysis discussed above to an existing ledger — the Cardano ledger with smart contract integration [5]. We note that a component of the ledger, called pointer addresses, is a thread which is update-deterministic, but not order-deterministic.

On the other hand, we conjecture that smart contracts implemented as state machines [3] constitute threads. Smart contract validation has been shown to be deterministic according to a notion of determinism specific to the Cardano ledger presented in [5]. Work remains to formally demonstrate that contracts do indeed constitute threads, and that that the definition of determinism in [5] is equivalent to the order-determinism we present in this work.

Future work As part of future and ongoing work, we intend to continue using the mathematical tools we discussed here to investigate ledger determinism. We hope to formulate a local, as opposed to trace-based, characterization of determinism, as well as characterize transaction validity in deterministic ledgers.

References

1 Florian Bridoux, Maximilien Gadouleau, and Guillaume Theyssier. Commutative automata networks. In Hector Zenil, editor, *26th International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA)*, volume LNCS-12286 of *Cellular Automata and Discrete Complex Systems*, pages 43–58, Stockholm, Sweden, August 2020. Springer International Publishing. URL: `https://hal.archives-ouvertes.fr/hal-02548573`, `doi:10.1007/978-3-030-61588-8\_4`.

2 Yufei Cai, Paolo G. Giarrusso, Tillmann Rendel, and Klaus Ostermann. A theory of changes for higher-order languages: incrementalizing λ-calculi by static differentiation. In Michael F. P. O'Boyle and Keshav Pingali, editors, *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, United Kingdom - June 09 - 11, 2014*, pages 145–155. ACM, 2014. `doi:10.1145/2594291.2594304`.

3 Manuel M. T. Chakravarty, James Chapman, Kenneth MacKenzie, Orestis Melkonian, Michael Peyton Jones, and Philip Wadler. The extended UTXO model. In *Financial Cryptography and Data Security - FC 2020 International Workshops, AsiaUSEC, CoDeFi, VOTING, and WTSC, Kota Kinabalu, Malaysia, February 14, 2020, Revised Selected Papers*, volume 12063 of *LNCS*, pages 525–539. Springer, 2020.

4 Leslie Lamport. Generalized consensus and paxos. 2005.

5 Polina Vinogradova and Andre Knispel. A formal specification of the Cardano ledger integrating plutus core. Technical report, IOG, 2021. Available at `https://hydra.iohk.io/build/14336115/download/1/alonzo-changes.pdf`.