Formalizing IOTA Extended UTXO in Isabelle

Edvardas Dlugauskas, Karolis Petrauskas

Vilnius University, Faculty of Mathematics and Informatics
Didlaukio str. 47, Vilnius
edvardas.dlugauskas@mif.vu.lt, karolis.petrauskas@mif.vu.lt

Abstract. The IOTA Extended UTXO (IOTA EUTXO) model extends the UTXO blockchain to include features like smart contracts and non-fungible tokens. In this work, we show that the IOTA EUTXO model maintains the base correctness properties of the UTXO model while extending it with extra functionality. We achieve this by specifying and verifying the essential concepts of the base UTXO model and the extensions proposed by IOTA using the Isabelle proof assistant. The specification is designed to be modular and extensible, meaning it can be used as a foundation for further research of the UTXO and IOTA EUTXO models.

Keywords: IOTA, UTXO model, EUTXO model, formal verification, Isabelle, formal methods.

1 Introduction

A blockchain is a decentralized digital ledger that records transactions in a way that is transparent and immutable. It uses accounts and tokens to represent digital asset ownership or rights, allowing peer-to-peer transactions without a trusted third party. The ledger in a blockchain network is usually implemented in one of two ways: using the Unspent Transaction Output (UTXO) model or the Account model. In the UTXO model, used by blockchains like Bitcoin, the ledger is represented as a set of unspent transaction outputs, referred to as just outputs in short [1]. Transactions consume outputs from the ledger as inputs and generate new outputs. This makes it possible to verify and process transactions which use different outputs as inputs independently. Subsequently, the UTXO model is known for its ability to allow parallel transaction processing, which improves scalability [2]. In contrast, the Account model, adopted by blockchains such as Ethereum, simplifies the ledger to a set of account balances [3]. Transactions adjust these balances directly. While more straightforward, this model requires transactions for the same account to be processed sequentially, which limits scalability [4].

DOI: https://doi.org/10.15388/LMITT.2024.3

In the UTXO model, digital assets (tokens), are represented using immutable outputs. The outputs are owned by actors, who can perform transactions, such as sending some amount of tokens to another actor. Crucially, each transaction irreversibly consumes its input outputs, generating new outputs to represent the transferred tokens. Every actor's owned amount of tokens at a given time can be calculated by taking the sum of tokens in all of the outputs owned by the actor. In the UTXO model, the current state is a set of all of the unspent transaction outputs. In other words, the UTXOs form a directed acyclic graph, and the current state is the set of all of the leaves of this graph [2].

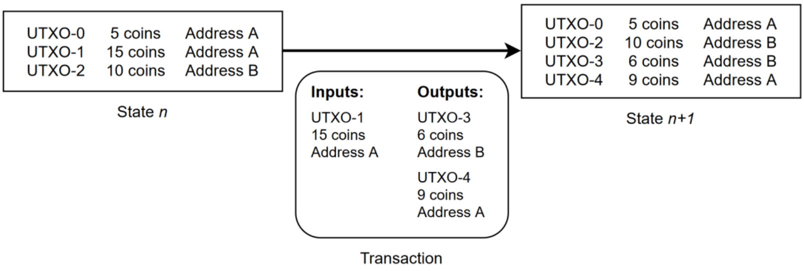

Figure 1. An illustration of a transaction in the UTXO model. In the UTXO model, the current state is a set of all of the unspent outputs.

Figure 1 illustrates a transaction in the UTXO model. In State *n*, we see three outputs: UTXO-0, UTXO-1, and UTXO-2, each associated with a certain number of tokens and an owning address (Address A or Address B). A transaction takes place where UTXO-1 is used as an input to create two new outputs: UTXO-3 and UTXO-4, which are then owned by Address B and Address A respectively. This transaction results in the subsequent State *n+1*, which now includes the two new outputs along with the unchanged UTXO-0 and UTXO-2 from the previous state.

The UTXO model supports some basic validation rules for output spending conditions. One of the most common conditions is the verification of ownership by the inspection of the digital signature on the output. However, there is a need for more programmable blockchain logic [5]. To address this, the concept of a smart contract can be used.

In general terms, a smart contract is a protocol for verifying and enforcing contracts on a blockchain [6]. A smart contract is stored on the distributed ledger, it inspects the state of the ledger, maintains and modifies its internal state, and performs actions such as creating new transactions. Smart contracts rely on more complex validation rules that are not supported by the base UTXO model. Due to the complexities of implementing a smart contract, the more straightforward account model is usually used instead [7].

One way to introduce expressive smart contracts while maintaining the semantic simplicity of the UTXO model is to implement the validation logic on the outputs. Subsequently, as the UTXO model is stateless, a smart contract's transactions would be forced to include any state information in the outputs themselves, introducing complexity to the model [5].

IOTA is a blockchain based on the UTXO model that started out with the aim of powering high throughput applications with low-price transactions [8]. TIP-18[1] is a design document that describes extensions to the IOTA UTXO model to add support for features such as NFTs and smart contracts. We refer to this proposed model as IOTA Extended UTXO (IOTA EUTXO) model.

The IOTA EUTXO model extends the traditional UTXO model. The goal of the IOTA EUTXO is to add the functionality of smart contracts while maintaining the base UTXO model's advantages. This is achieved by appending additional data fields and extending the validation logic in the outputs. Thus, the EUTXO model allows for more complex transactions and behaviors without limiting the model's scalability [9].

Correctness is crucial to blockchain technologies as every processed transaction, whether correct or not, is permanent. This means that any oversight or vulnerability in the transaction processing logic can be impossible to revert [10]. Subsequently, the complexity of the changes proposed in the IOTA EUTXO design document raises the question of IOTA EUTXO model's correctness.

Formal methods are a set of techniques to accurately specify and verify software systems [11]. By applying formal methods to the verification of blockchain protocols and smart contracts, we can identify any incorrect behavior of the system at an early stage [12]. In the case of IOTA EUTXO model, formal methods can be used to prove the correctness of the proposed model [13].

1 https://github.com/lzpap/tips/blob/master/tips/TIP-0018/tip-0018.md

The formal verification of blockchain models or their smart contracts often involves using a proof assistant [14]. Isabelle is a collection of tools that allow formally verifying specifications using higher-order logic [15]. It is currently one of the more popular formal verification tools in academia due to its intuitive development environment and use of powerful provers. As such, Isabelle is a solid choice for specifying and verifying both the base UTXO model and IOTA's EUTXO model.

While there have already been attempts to formalize the UTXO model, some of them using Isabelle, the results of these attempts are difficult to reuse for formalizing the IOTA EUTXO. Many of the formalizations, such as the Cardano UTXO specification[2], are missing an accompanying paper, which complicates further analysis of the specification, and do not explicitly consider the possibilities of extending the model. Other specifications, such as the mathematical specification of the UTXO model by Gabbay et al., use manual proofs [16]. Furthermore, the abstract nature of the models does not address the concern of their real-life applicability. Thus, a new formalization of the UTXO and IOTA EUTXO is required.

In this paper, we formalize the IOTA EUTXO model using Isabelle. We demonstrate a way to represent the essential entities and properties of the UTXO model in a modular way using Isabelle's syntax, including the locale construct. We then build on the UTXO model by formalizing a subset of the IOTA EUTXO's functionality. We show that the IOTA EUTXO model maintains the base UTXO model's properties while allowing for more complex workflows. By splitting the specification into implementation and abstract parts, we ensure that it is both theoretically sound and practically feasible. Our work provides a solid foundation for future UTXO and IOTA EUTXO model research by creating reusable components in Isabelle.

2 Formalizing the UTXO Model in Isabelle

Our formalization of the UTXO model in Isabelle uses a two-layered approach, differentiating an abstract and a implementation specification. The abstract layer represents the core properties and operations of outputs, transactions, and the ledger without tying them to specific data types – relationships and properties are represented using generic predicates

2 https://github.com/input-output-hk/cardano-ledger-high-assurance/blob/master/Isabelle/UTxO/UTxO.thy

instead. This allows for the verification of the UTXO model's properties in a generic manner, ensuring any valid concrete implementation will inherit these properties.

When reasoning about the abstract model, we found the Isabelle locale construct to be very useful in ensuring a modular and extendable specification. A locale in Isabelle is a collection of parameters and assumptions that provide a context for proving theorems. To be more precise, locales are a way to define abstract contexts and structures, which can be instantiated later with specific types, functions, or relations. They provide a mechanism to reason about abstract properties and assumptions, prove theorems in a generic context and reuse the results in specific instances.

$$\forall x_1, \dots, x_n.[(A_1; \dots; A_m) \Rightarrow C] \tag{1}$$

In Eq. 1 parameters x_1 to x_n are fixed, assumptions A_1 to A_m are made, and the conclusions C are implied. When writing Isabelle code, C would correspond to the proofs for lemmas and theorems that can be proven inside the context of the locale. A locale can then be instantiated by satisfying its parameters and assumptions using a specific type by using the interpretation mechanism, which allows us to reuse the proven properties of the locale.

The abstract model focuses on the UTXO model's essential entities and properties. Using Isabelle locales, we model basic entities like outputs, transactions, and the ledger, each with its own essential properties and operations. For instance, the *basic_output* locale guarantees that each output possesses a non-zero amount, while the basic transaction locale ensures the conservation of total token amount in a transaction.

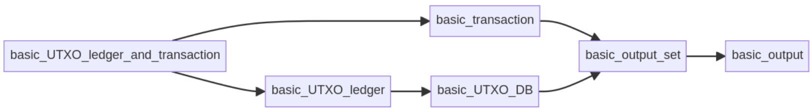

Figure 2. The locale structure in the abstract UTXO model. Six distinct locales are represented as nodes, the relationships between them are depicted with directed edges.

Figure 2 depicts the hierarchical structure of locales in the abstract UTXO model as used in formal verification within the Isabelle proof assistant framework. It illustrates how six distinct locales interconnect to model the UTXO model, with each node representing a locale and the edges indicating the dependency and extension relationships between them. The fundamental locale is *basic_output*, which defines individual UTXOs. It is used by *basic_output_set*, representing a collection of UTXOs. This, in turn, is referenced by the *basic_transaction_locale*, which models the transaction mechanism. The *basic_UTXO_ledger* locale relies on *basic_UTXO_DB*, which itself references *basic_output_set*. Both *basic_transaction* and *basic_UTXO_ledger* are used by the *basic_UTXO_ledger_and_transaction* locale, which encapsulates the ledger's state and a valid transaction, allowing us to reason in terms of the current and the subsequent ledger's state.

In contrast, the implementation layer offers a concrete example implementation of the UTXO ledger, defining specific data types for outputs, transactions, and other essential entities, as well as functions and predicates to model ledger updates and transaction validity. By mapping these concrete types to their abstract counterparts and verifying that the abstract model's assumptions still hold, we demonstrate that the implementation adheres to the desired properties of the UTXO model.

By using Isabelle's interpretation mechanism, we link the implementation model's concrete entities to the abstract model's locales and assumptions. This validates the implementation against the abstract specification and ensures the inheritance of all proven properties from the abstract model. Thus, we establish the correctness of our UTXO model implementation, by ensuring it is both theoretically sound and practically viable.

3 Formalizing the IOTA Extended UTXO Model in Isabelle

The IOTA EUTXO design document describes several new output types such as alias output and foundry output. An alias output is an output representing smart contract invocation chain accounts that can process requests and transfer funds. A foundry output is an output that contains the state of and manages user-defined native tokens [17]. To support these new output types, the ledger has to have some characteristics of a state machine.

For an output to function as a state machine, the state of the output must be moved forward when it is consumed as an input. In the UTXO model, the input outputs are essentially burned and only the value amount is distributed among the outgoing outputs. Subsequently, IOTA proposes an extension to the validator called a chain constraint. The chain constraint allows the transfer of the output state machine state encoded in the new additional fields on the output across transactions. The alias output and foundry output utilize this chain constraint to transfer the states of the alias state machine and foundry state machine respectively. The chain constraint describes validation rules that ensure that the state is created, modified, and deleted in a correct manner. For example, the chain constraint ensures that once an alias is created, it has a continuous existence across the ledger until explicitly deleted and removed from the ledger.

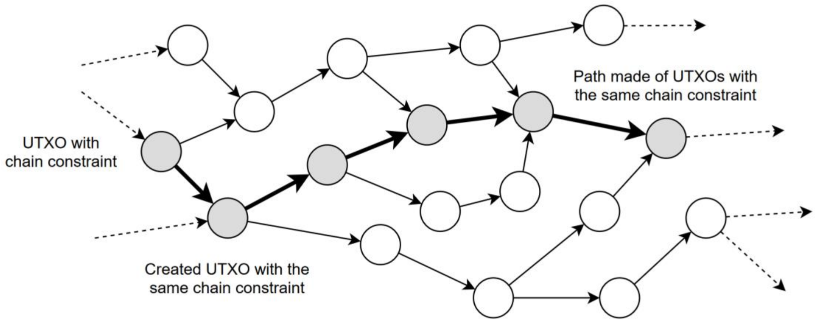
Path made of UTXOs with the same chain constraint
UTXO with chain constraint
Created UTXO with the same chain constraint

Figure 3. A path of consumed outputs with the same chain constraint forms a chain. Arrows indicate the creation of an output and the circles are the transactions; gray circles are transactions that contain a chain constraint.

Figure 3 visualizes the concept of a transaction chain that uses the chain constraint. In this representation, each circle symbolizes a transaction, with the gray circles representing transactions that include an output with a chain constraint. The arrows between the circles represent the use of an output from one transaction in the next. A path made up of outputs linked by the same chain constraint is highlighted. This chain begins with the creation of an output with a chain constraint and terminates when the output containing this constraint is spent without creating a new one, which ends the sequence.

Our formalization of the UTXO model in Isabelle uses the same two-layered approach used for the base UTXO model, with an abstract and a implementation specification. The IOTA EUTXO formalization builds upon the locales of the basic UTXO model.

In the abstract model, the details of the alias and foundry outputs are added. Notably, the base output and the additional state fields in an alias or foundry output are independent in terms of the operations and the properties of the UTXO model. Properties and operations from the base model only reference fields of the base model. Analogously for alias and foundry outputs. Thus, we have modeled them as separate ledgers. For example, an alias ledger is an image of the blockchain ledger containing only the alias parts of the outputs; an alias ledger contains details of only the alias outputs and validates logic related only to alias output functionality. This approach ensures that our specification remains modular and reusable.

In the implementation model, we have opted to use a single ledger to represent all of the output types, which better mimics a possible real-world implementation. Instead, we define the ledger as a set of a sum type with three possible constructors: basic output, alias output and foundry output. The basic output contains only the base output fields, while alias and foundry contain both the basic output fields and the alias and foundry fields respectively. This allows us to map the implementation to the abstract specification – an alias ledger is just the image of the implementation ledger which takes all of the alias field parts of the outputs in the ledger.

Using the Isabelle locale interpretation feature, we link the IOTA EUTXO implementation model's concrete entities to both the base output and the IOTA EUTXO abstract model's locales and assumptions. Subsequently, we establish the correctness of the implementation model in the context of both the base UTXO and the IOTA EUTXO models' invariants and properties.

4 Formalization Results

In our work, we not only formalized the models, but also verified some of their properties using Isabelle automated provers.

The UTXO model has several essential properties that we have verified:

- Constant Supply: the sum of unspent outputs in the ledger must be constant.
- Unspent Output Consumption: an output can be consumed only if it is a part of the current ledger state and this output will not be present in the subsequent ledger state.
- No Double Spending: an output can only be consumed by a single transaction.

In the IOTA EUTXO model, we have verified all of the base UTXO model's properties in addition to the chain constraint for the alias output:

- Continuity of Alias (Chain Constraint): once an alias is created, it has a continuous existence across the ledger until explicitly deleted and removed from the ledger.

The proof process for verifying these properties in Isabelle involved several steps. For each property we aimed to verify, we started by formulating a theorem definition for it using Isabelle. This required translating informal descriptions of UTXO model's behavior into precise, logical statements using Isabelle's syntax. We then used Isabelle's automated proof search tools and manual proof strategies to construct a proof for each theorem. Finally, we used Isabelle's automated provers to ensure that the proof was sound.

To demonstrate the verification of one specific property in more detail, let's consider the constant supply property of the UTXO model. This property ensures that the total sum of tokens across all unspent outputs remains unchanged by the application of transactions, assuming no new tokens are minted or existing tokens are destroyed outside of transactions.

We first formally defined the *sum_amount* function in Isabelle, which calculates the total sum of tokens in a given set of outputs. The *constant_supply* theorem was then stated as:

$$\mathit{sum\_amount}\ \mathit{DB} = \mathit{sum\_amount}\ (\mathit{apply\_transaction}\ \mathit{DB}\ \mathit{tx}) \tag{2}$$

In Eq. 2 we assume that *tx* is a valid transaction in the ledger *DB*.

We then constructed a proof by interacting with Isabelle's automatic proof search functionality. The proof uses the subproofs for the facts that the amount of tokens in the inputs and outputs of a valid transaction is the same, and that, in terms of tokens, applying a transaction is equivalent to subtracting the amount of tokens in the inputs and adding the amount of tokens in the outputs. Finally, the Isabelle automated provers verified our proof to demonstrate its soundness.

5 Conclusions

We aimed to formalize the IOTA EUTXO model by utilizing the Isabelle formal verification tool and verify it using Isabelle's automated prover. Our formalization showed that the IOTA EUTXO model not only retains the base correctness properties of the base UTXO model including no double spending, constant supply, and unspent output consumption, but also supports additional properties which are specified in the IOTA EUTXO design document, such as the chain constraint.

We have used a two-layered approach in our formalization to ensure that our model is both theoretically robust and practically applicable. The abstract layer allowed us to verify the UTXO model's core properties in a generic way, while the implementation layer provided a concrete example that adheres to the verified properties, demonstrating the practical viability of our formalization.

We have presented a model that is not only modular but also extensible. Our approach to formalization, which uses Isabelle locales, provides flexibility in specifying the model by allowing easier future extensions or modifications. This is crucial for keeping the model relevant as distributed ledger technologies continue to evolve. We believe that this approach offers a good foundation for further research and developments in the field of UTXO blockchain technologies.

References

[1] Almeida, J. B., Frade, M. J., Pinto, J. S., and De Sousa, S. M. (2011). Rigorous software development: an introduction to program verification, volume 1. Springer.

[2] Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Gollamudi, A., Gonthier, G., Kobeissi, N., Rastogi, A., Sibut-Pinote, T., Swamy, N., and Zanella-Béguelin, S. (2016). Short paper: Formal verification of smart contracts. In Proceedings of the 11th ACM Workshop on Programming Languages and Analysis for Security (PLAS), in conjunction with ACM CCS, pages 91–96.

[3] Brünjes, L. and Gabbay, M. J. (2020). UTxO- vs account-based smart contract blockchain programming paradigms. In International Symposium on Leveraging Applications of Formal Methods, pages 73–88. Springer.

[4] Chakravarty, M. M., Chapman, J., MacKenzie, K., Melkonian, O., Jones, M. P., and Wadler, P. (2020). The extended UTXO model. In International Conference on Financial Cryptography and Data Security, pages 525–539. Springer.

[5] Chakravarty, M. M. T., Chapman, J., MacKenzie, K., Melkonian, O., Müller, J., Peyton Jones, M., Vinogradova, P., and Wadler, P. Native custom tokens in the extended UTXO model. In Margaria, T. and Steffen, B., editors, Leveraging Applications of Formal Methods, Verification and Validation: Applications, Lecture Notes in Computer Science, pages 89–111. Springer International Publishing.

[6] Delgado-Segura, S., Pérez-Sola, C., Navarro-Arribas, G., and Herrera-Joancomartí, J. (2018). Analysis of the bitcoin UTXO set. In International Conference on Financial Cryptography and Data Security, pages 78–91. Springer.

[7] Fatkina, A., Iakushkin, O., Selivanov, D., and Korkhov, V. (2019). Methods of formal software verification in the context of distributed systems. In International Conference on Computational Science and Its Applications, pages 546–555. Springer.

[8] Gabbay, M. J. (2022). Algebras of UTxO blockchains. Mathematical Structures in Computer Science, pages 1–56.

[9] Liu, Y.-C., Fang, J., and Liang, J.-W. (2019). Account-wise ledger: A new design of decentralized system. Github. https://github.com/ECS-251-W2020/final-project-triple-l-group/blob/master/Thesis/Account-Wise%20Ledger.pdf

[10] Melkonian, O. (2019). Formalizing Extended UTxO and BitML Calculus in Agda. Master's thesis. Utrecht University Student Theses Repository Home. https://studenttheses.uu.nl/bitstream/handle/20.500.12932/32981/thesis.pdf

[11] Nipkow, T., Paulson, L. C., and Wenzel, M. (2002). Isabelle/HOL: a proof assistant for higher-order logic, volume 2283. Springer Science & Business Media.

[12] Popov, S. and Lu, Q. (2019). Iota: feeless and free. IEEE Blockchain Technical Briefs.

[13] Ribeiro, M., Adão, P., and Mateus, P. (2020). Formal verification of ethereum smart contracts using Isabelle/HOL. In Logic, Language, and Security, pages 71–97. Springer.

[14] Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., and Wang, F.-Y. (2019). Blockchain-enabled smart contracts: architecture, applications, and future trends. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49(11):2266–2277.

[15] Wood, G. et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151(2014):1–32.

[16] Zhang, J., Tian, R., Cao, Y., Yuan, X., Yu, Z., Yan, X., and Zhang, X. (2021). A hybrid model for central bank digital currency based on blockchain. IEEE Access, 9:53589–53601.

[17] Zheng, Z., Xie, S., Dai, H., Chen, X., and Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. In 2017 IEEE international congress on big data (BigData congress), pages 557–564. IEEE.