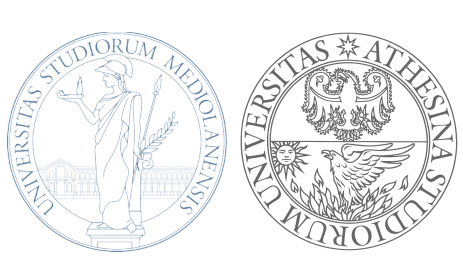
Design and Analysis of Tokenomics Processes

Simone Valentini[1], Irene Domenicale[2], and Elvinia Riccobene[1]

[1] Università degli Studi di Milano, Milan, Italy
[2] University of Turin, Turin, Italy



TOKENOMICS

Tokenomics refers to the economic analysis of cryptographic tokens, encompassing supply management (emission and burning mechanisms), distribution strategies, value determination, and functional utility within blockchain-based systems [1]

[1] Young Sook Kim, Seng-Phil Hong, and Marko Majer. Validation of value-driven token economy: focus on blockchain content platform. Future In- ternet, 16(5):178, 2024.

RELATED WORK

Alredy existing aproaches allow to model and analyze an economy textual languages. CadCAD [1], CasCAD2 [2], Tokenomics-Simulator [3] and Machinations [4]

- Require **technical** expertise
- Not specific for tokenomics
- No automatic **translation**

[1] https://cadcad.org/

[2] X. Liang et al., "From cadCAD to casCAD2: A Mechanism Validation and Verification System for Decentralized Autonomous Organizations Based on Parallel Intelligence," in IEEE Transactions on Computational Social Systems, vol. 11, no. 2, pp. 2853-2862, April 2024.

[3] https://crates.io/crates/tokenomics-simulator/0.1.1

[4] https://machinations.io/

TATAMI

TATAMI is a platform-independent workflow-based visual modeling language specifically tailored for tokenomics

Canvas

- **Interactive workspace** for arranging blocks
- Blocks interconnected in **token flows**

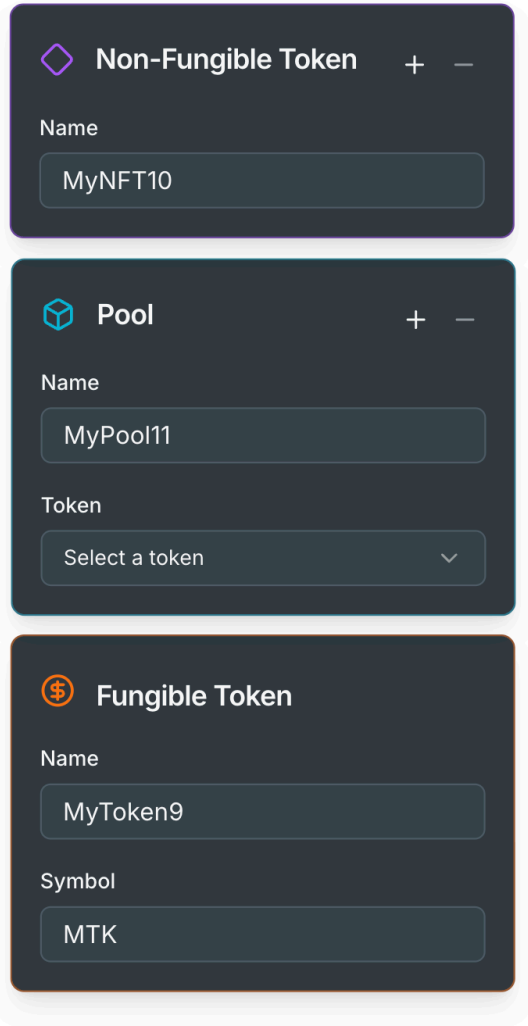
Model State

- Aggregates **inputs, variables, mappings, and constants**
- **Dynamically** read and modified by flow blocks

Independent Elements

Elements that need to be managed throught token flows.

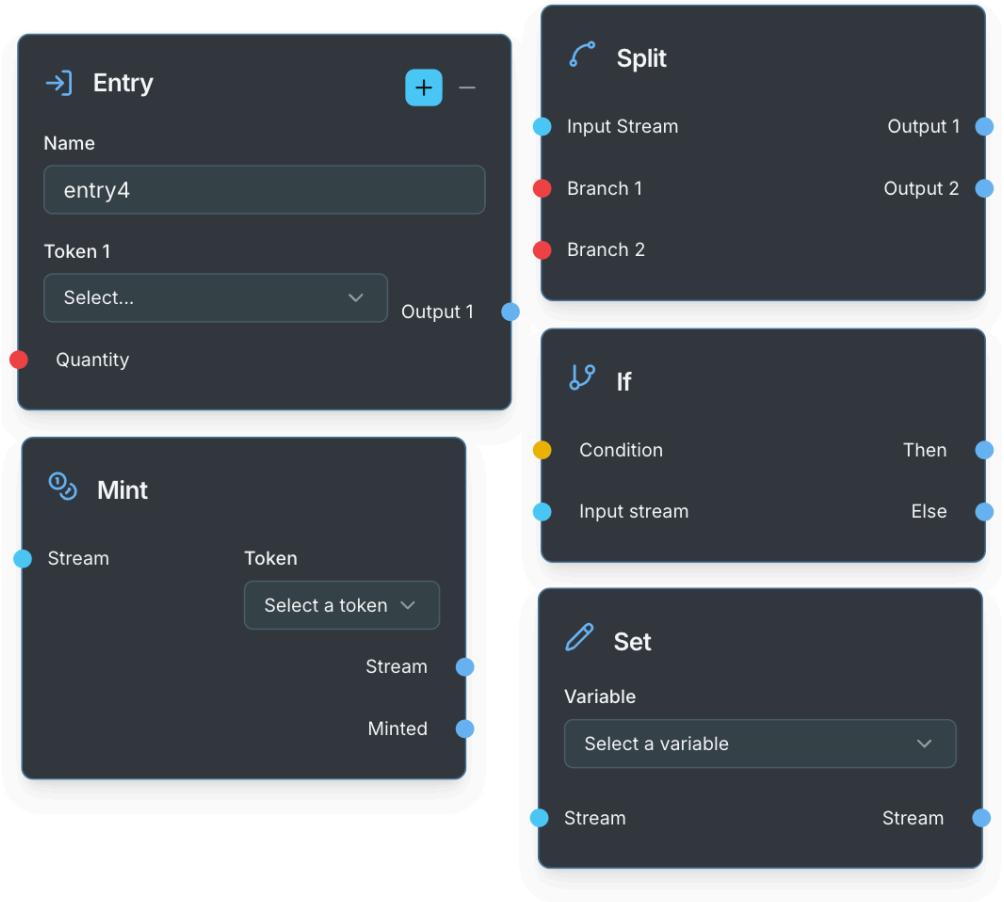
- **Fungible Tokens**: Identical tokens
- **NFTs**: Unique assets with distinct identity and custom metadata
- **Pool**: Key-based containers for accumulating tokens



Flow Blocks

Blocks interconnected through token streams.

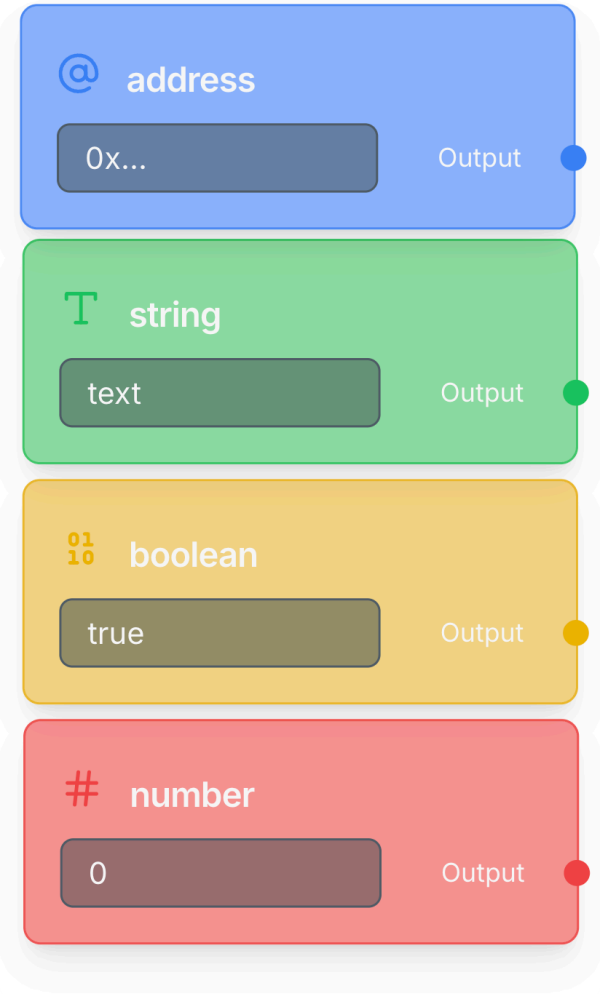
- **Entry**: Mandatory starting point for every workflow
- **Withdraw / Deposit**: Interaction with Pools
- **Mint / Burn / Transfer**: Token lifecycle management
- **If / Split / Join / Set**: Token streams and model state manipulation



Type Blocks

Blocks to cast some text or reference to the model state to a specific type.

- **Numeric**: Math expressions
- **Boolean**: Boolean expressions for conditional blocks
- **String**: Value for names, or descriptive attributes
- **Address**: Identifiers for accounts interacting with the tokenflow



Model State Component

The Model State manages the data layer of the system, including both generated and custom elements.

Auto-generated

Elements created by the system:

- **Sender** and **BlockNumber**
- **FT/NFT** attributes
- **Pool** balances

Custom Elements

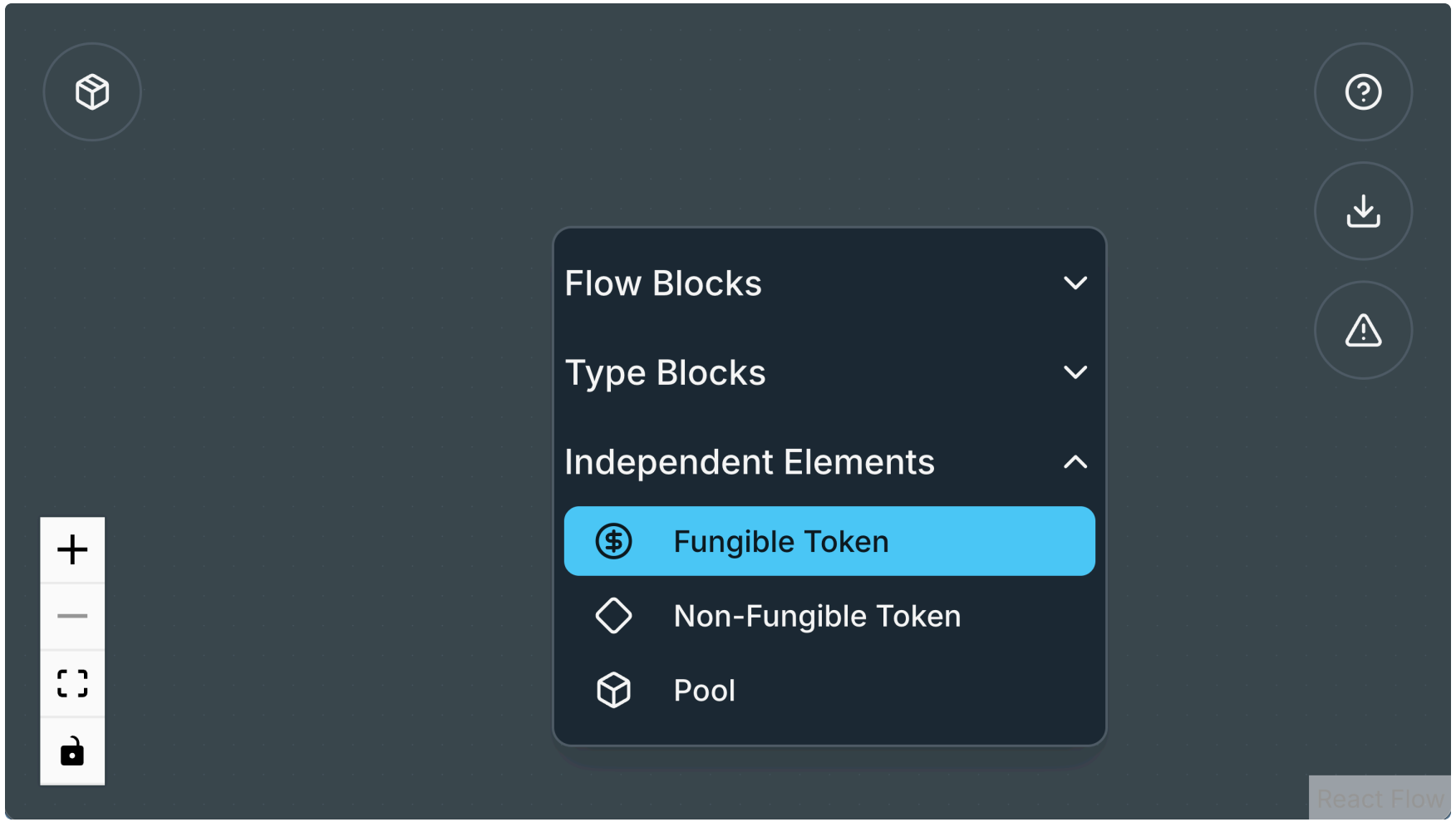
Defined by the user:

- **Variables**
- **Constants**
- **Inputs**

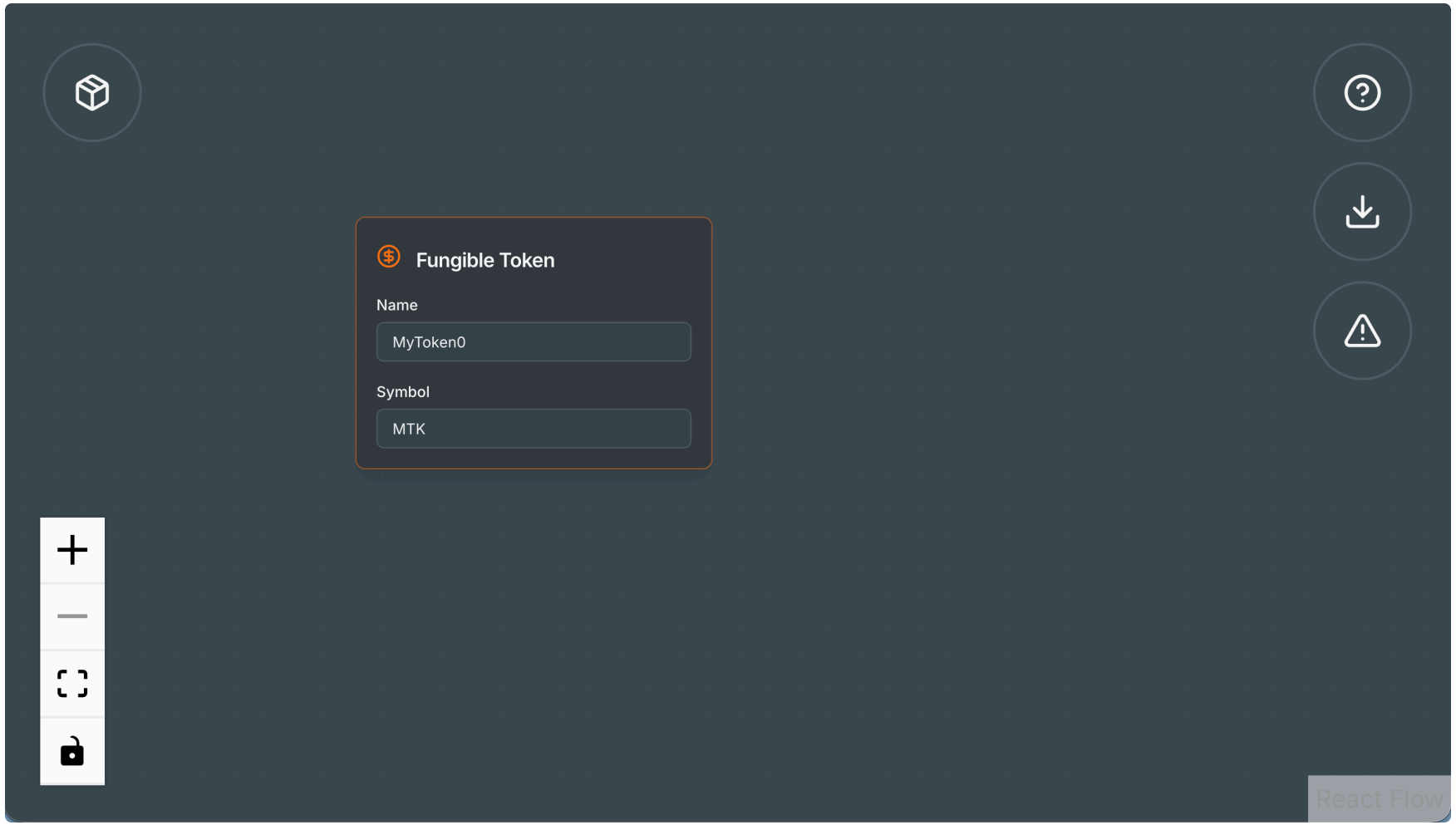
The Editor [1]

[1] The editor is available at https://github.com/simone20a/TATAMI

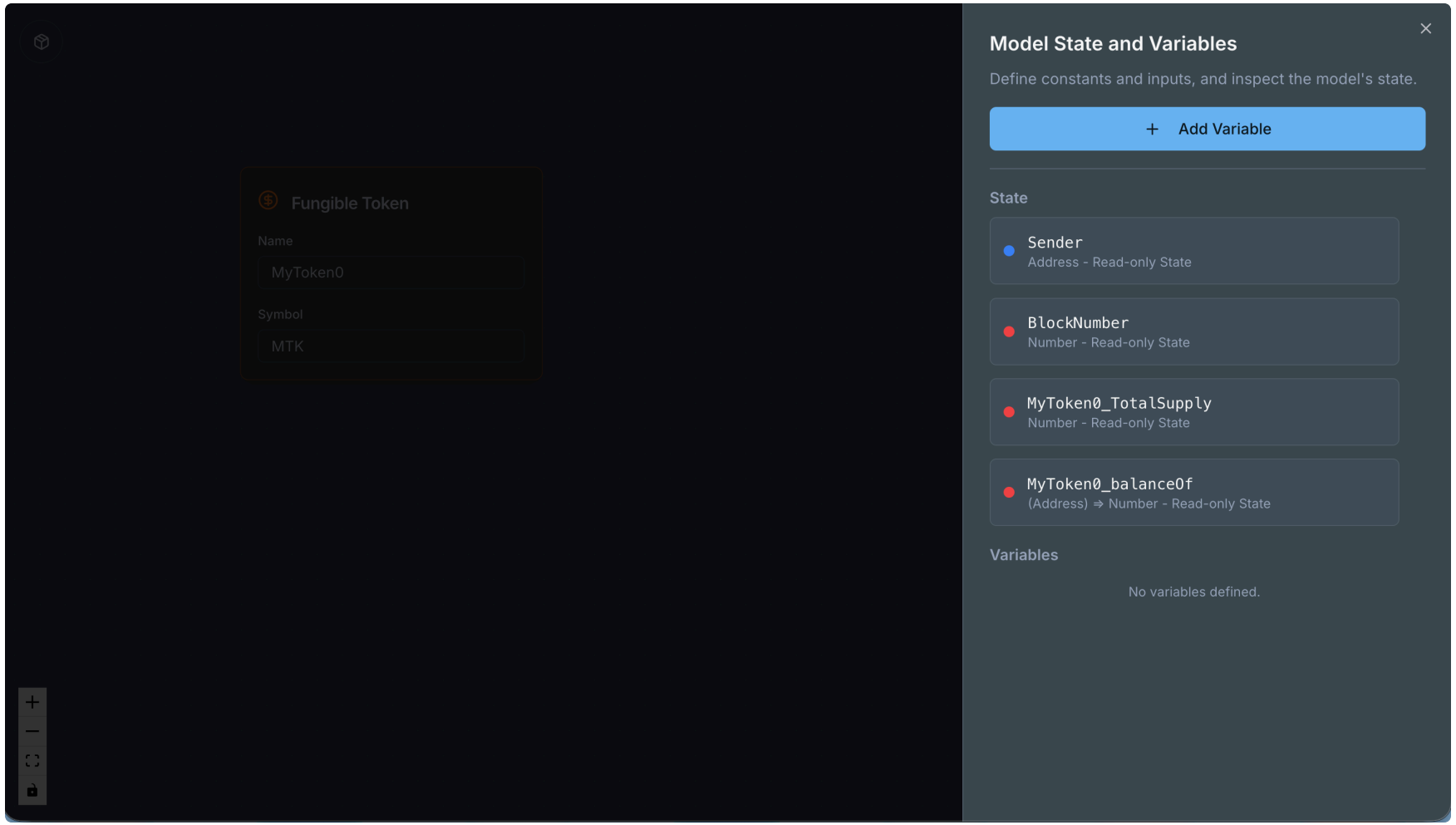
The Editor


Flow Blocks
Type Blocks
Independent Elements
Fungible Token
Non-Fungible Token
Pool
React Flow

The Editor



The Editor


Fungible Token
Name
MyToken0
Symbol
MTK
Model State and Variables
Define constants and inputs, and inspect the model's state.
+ Add Variable
State
Sender
Address - Read-only State
BlockNumber
Number - Read-only State
MyToken0_TotalSupply
Number - Read-only State
MyToken0_balanceOf
(Address) ⇒ Number - Read-only State
Variables
No variables defined.

GOALS

Short Term

- Provide a **formal semantics** to the language using **Abstract State Machines** (ASM)
- Create a **compiler** to ready-to-deploy Solidity code

Long Term

- Explore application of ASMs in **formal verification** of TATAMI models to check:
  - Model Correctness
  - Oversupply Prevention
  - Inflation Spirals Avoidance
  - Burn-Emission Equilibrium

    ...

Thanks