Agorata

Lev Chizhov[1,2]

@ennucore

July 17, 2022

Abstract

Agorata is an implementation of the idealized economic agent, or, specifically, an aggregator of smart contracts. It combines one or more contracts proposed by the members of the network in a deal which can be evaluated as profitable for Agorata with low risks. Using this approach, Agorata can provide infrastructure for loans, flashloans, bets, derivatives, bridging between chaiNs, and many more financial instruments.

Contents

I. Introduction and overview

There is a variety of standard financial organizations that provide their capital for some simple purely financial deals: banks, brokers, gambling institutions, betting institutions and many more. Most of them have their own DeFi counterparts, as well as some services that are only possible on the blockchain — for example, flashloans.

[1] Moscow Institute of Physics and Technology
[2] Ludwig-Maximilians-Universität München

All of these organizations can be generalized to an economic agent that has a utility function and who takes the deal if and only if it is the most beneficial in terms of this function. These particular implementations, however, only consider a very limited subset of deals. Also, all of them work only with purely financial deals which include not more than 3 parties[1], including the financial organization.

Technologies of decentralization and smart contracts in particular provide the foundation for a much more fundamental entity — one which goes far beyond standard simple templated purely-financial deals. Instead, this entity can be the implementation of the general economic agent. It can analyze deal proposals and combine them into a very complex deal, which it will evaluate and participate in. As a result, this entity can serve as a liquidity provider, loan broker, a service for finding a counterparty. It is worth noticing that as the blockchain technologies spread into multiple fields (not only financial), a service like that can become much more versatile and participate in deals including various assets (domain names, art, computational resources are the examples of what can be possible now, future applications will go much further).

The goal of this whitepaper is to present such an entity — Agorata.

II. Decision-making algorithm

In this chapter, the strategy for making decisions will be described.

i. Deal representation

A deal proposal is made by the user in the form of a smart contract. A smart contract is considered as an entity with which other entities (users, smart contracts) can interact via messages[2]. For a state of the contract we can determine the messages that can be sent to the contract and for each of them — the response messages and the next state. The deal is represented as a tree with states as leafs and actions (messages) as edges. Depending on the actions, the deal has a total outcome — the number of tokens that were gained/spent by the agent.

Each action (message) has parameters θ, including:

1. t — the time at which the message is sent.
2. s — the sender of the message.
3. tokens sent with the message.
4. any other possible information.

The parameter space for each action can be constrained — the smart contract can reject some of the messages. For instance, this can lead to the entire space consisting of a single element (the contract rejects everything except for one specific message, e.g. a specific person sending a specific amount of tokens).

The main parameters of the state are the financial outcomes for the smart contracts. The state is written as a function of all parameters of all actions in a formal language.

[1] An example of such a deal with three parties is a stanard deal on a financial market.

[2] The message concept used here is from The Open Network (TON). A message can include tokens, commands, information, code.

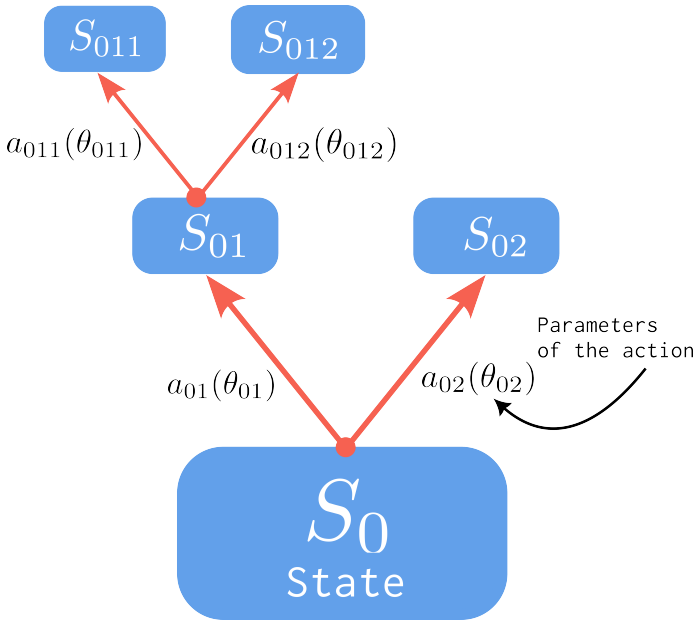

Figure 1: *Deal representation.*

ii. Agent parameters

1. T_0 is the maximum deal duration — i.e., the horizon after which the benefits of the deal are not considered.

2. τ is the charachteristic time of discount — e.g., $\exp(\frac{1\,year}{\tau}) - 1$ is the minimal yearly rate for a loan.

Given these parameters, the value function $V((x_i), (t_i))$ (or $V(S)$) can be determined. It takes the sequence of values (x_i, t_i), where x_i are the incoming/outcoming tokens from the agent.

iii. Deal evaluation

The agent considers the worst case of the deal from the perspective of game theory: the best (from the perspective of the value function) actions of the agent and the "worst" actions of the counteragents. This sequence of events is the run through the value function in order to make a decision.

Thus, a deal is considered profitable for the agent $(A) \iff p(S_i) \iff$

$$V(S_i) > 0 \vee (\forall a_{ij}(V(S_{ij}) > 0 \vee a.s = A) \wedge a_{ij} : V(S_{ij}) > 0)$$

The algorithm starts at the node S_0, then considers all the possible actions (including the abscence of one) and calls the algorithm at S_{0i}. If there is any action of the counteragent that makes the contract non-profitable for the agent, the deal is considered non-profitable. If there is no possible action and the state includes profit for the agent, the deal is considered acceptable.

iv. Example

III. Contract evaluation algorithm

In the previous section of the article we specified how the decisions are made. One of the most important parts of the process is finding out whether there exists any action that makes the deal profitable or unprofitable. It is also important to know which actions the agent should make to maximize its output. How do we do that?

The state parameters are represented as functions of action parameters expressed in a formal language.

IV. Applications

i. Flashloans

ii. Loans

iii. Bridges

iv. Bets

v. More complex examples

V. Conclusion