PLURAL: TOWARDS A DECENTRALIZED FINANCE AND INSURANCE ECOSYSTEM

THE PLURAL FINANCE TEAM

Abstract. Decentralized Finance (DEFI) opens the frontier for using smart contracts to build collaborative and inclusive financial services. The difference with traditional, formal, legally binding contractual agreements is that smart contracts execute agreements based on their underlying programming. Further, when these contracts are on a public blockchain or other public distributed ledger, their behavior is independently verifiable and auditable. DEFI offers opportunities at a scale that is genuinely global, inclusive, open, and permissionless: Any user with a crypto wallet can interact with a DEFI contract and leverage the opportunities and risks offered therein. For those that would re-imagine finance, DEFI provides the chance to create an alternative financial and financing narrative that is participatory and comparatively frictionless. One area where low friction scale can have an impact is insurance: At an estimated five trillion dollars annually, insurance is big and expensive. Insurance plays a significant role in the lives of most people in the world, either directly or indirectly or through the lack of access. This paper introduces the Plural Ecosystem, a proposed decentralized finance ecosystem that aims to build and scale insurance for Web3 users and products, and beyond.

1. The Nature of Insurance

Modern insurance comprises a contractual relationship between an end-user and an insurance provider. The end-user, later called a policyholder, engages with the insurance provider to pass their risk exposure to the insurance provider. The insurance provider, in turn, engages with the user to transfer risks and offer them coverage. The cost of this exchange is deemed a premium, and the contract between the parties is called a policy. The policyholder is guaranteed certain rights under the policy contract. The insurer is entitled to charge for these guarantees, usually on an ongoing basis: To maintain the agreement, policyholders commit to premiums, and insurers are required to honor valid claims on policies by providing policyholders with either benefits or indemnifications against losses, as stipulated within their contracts. In effect, the parties exchange risks, the insurer agrees to cover future liability risks for the policyholder; while the policyholder accepts the risk that the insurer may not pay out against future liabilities as required by the contract, for various reasons, including bankruptcy.

Date: 01 August 2023; v0.7.

Insurers diversify across policyholders by contracting with a spectrum of agents, on the premise that only some of those agents will be claimants on their policy agreements. From a social perspective, insurers play the role of transferring contingent costs from individuals to groups; lowering costs on average, across cohorts of policyholders.

The arrangement works perfectly when policyholders and insurers play by the 'rules' that may be both implicit and explicit. First, policyholders 'ought' not to file false claims, which the legal system deals with punitively in most countries. Secondly, insurers 'ought' to price policies fairly and honor claims timeously and reasonably. However, as humans, we do not always have each other's best interests at heart, and this type of contractual arrangement introduces principle-agent dilemmas: The insurer acts as an agent for the insured and, post payment, there may be no reason for the insurer to honor the contract between the parties.

For this reason, the modern insurance industry is heavily regulated. Insurance products expose the insurance underwriter to the risk of potential capital losses while simultaneously concentrating power in the hands of the insurer, who ultimately decides whether to settle or not to settle a given claim. On the underwriting side, re-insurers provide insurance on insurance, while on the policy side, independent bodies (including the courts) balance the concentration of power held by insurers.

For now, let us backtrack to the origins of insurance as a form of mutual protection within communities. The original 'mutual' form of insurance was a group of agents with similar exposure who banded together to mitigate their risk. In this case, they agreed to collect a pool of financial resources, understanding that funds from the pool would be allocated to help members recover from losses. In turn, fraudulent claims and mismanagement of the pool of funds are down regulated through social pressure. In small communities, everyone knows everyone else's business: The challenge is to scale this public model for information to a modern setting.

In either case, the unspoken question is whether having external insurance or self-insurance is better. The subtlety here is that there is no such thing as 'going without insurance': An economic agent without insurance has effectively accepted their (i) losses in advance or (ii) is willing to fund their own recovery after a loss.

We buy insurance to protect ourselves from an event that 'might' happen. The event occurs with some frequency amongst some cohort of similar agents: So, for example, people that own cars tend to have accidents on occasion which, in turn, means that all people that drive cars carry some fractional risk of having an accident. When multiple agents in a similar cohort agree to share the costs of a loss of an event with other members of the cohort, this diversifies the cost of

the loss amongst all members. Arguably, as long as the price to an individual of being a member in the 'paying cohort' is lower than the payout required when a loss event happens, there is a benefit to being externally insured. The additional factor here is the time value of money; regular short-term premiums tend to be smaller than opportunity cost of large payouts or borrowed capital required to cover sudden losses. And, some failures can have significant downside risks - it is too expensive not to be insured because the loss, when it happens, can cost everything: This motivates parents with young children to get insurance and find godparents, and climate activists to protest against oil companies: For them, the downstream costs are too high.

2. Insurance Models

The above discussion gives us many pointers on how to model insurance systems, and note that this is different from modelling insurance products and making risk assessments. Insurance systems are multiparty agreements which require resource transfer commitments, resource management commitments, claims validation systems, identity verification systems, and payment triggers. Moreover, these systems evolve in the sense that while the state of the system may be known for a given fixed instance, the state of the system may have been different in the past and may be different in the future.

2.1. **Insurance State Space.**

Here we describe a generic abstract state space for insurance style contracts. Our model of time is discrete, and it is assumed that events can be ordered so that notions of past and future are well defined.

2.1.1. *Time Parameter.* Let t be a discrete parameter and suppose that $t-1$ precedes t, and $t+1$ proceeds t. We can think of t as a counter for rounds in a discrete game; or blocks in a blockchain. If the clock is initialized at $t=0$, write $t \in I$ where $I = \{0, 1, 2 \ldots\}$ is a countable set. A fixed value of t gives us a snapshot into the state at a particular round. For our particular model, assume that no relevant activity happens between snapshots, or that changes to the state-space are accumulated and appended into the next time slot.

2.1.2. *Economic Agents.* Policyholders and insurers are economic agents in our system. They are decision makers that can affect the state locally, through their decisions. Suppose that there is universal set of economic agents S, and that every economic agent involved in our insurance system is a member of that set. We suppose that each agent in S can be identified uniquely, so that we can track their behaviour and their choices over time. In a blockchain context, agents are identified by their public addresses. For the agents in a particular round we write $S(t)$ or S^t. We suppose that once an agent joins their records are persistent

so $S^0 \subset S^1 \subset S^2 \ldots$ etc. Essentially, the namespace of each subsequent round contains the namespace of the previous rounds. The list of policyholders is a subset $H \subset S$, and the list of underwriters is a subset $U \subset S$. We may also consider a list of agents that are policy creators $D \subset S$ and claims evaluators $E \subset S$. A closed member-only insurance mutual where claims are voted on and policies are created by members is, for example, a special instance where $H \equiv U \equiv D \equiv E$. For individual members we write $s \in S$ and use lower case letters. If we want to emphasise the state of a particular subset say, $H \subset S$, at a particular time, we write, for example, $H^0, H^1, \ldots, H^t$ etc.

In this type of discrete game setting, the state of the system is an interplay between agent behaviour and the rules of the system: Agents can have rights and permissions, vote on modifications to the system parameters and rules, and own token balances. For our purposes, the system rules are encapsulated within smart contracts on an underlying blockchain or distributed ledgers. Using set theory in this context allows us to label and keep track of the different agent types and interactions.

2.1.3. *Token Pools.* We suppose there is a collection of token pools (accounts or addresses) such that each token pool can be assigned a balance. Let J be the set of all such pools. For an individual pool write $j \in J$. Economic agents are able to make deposits and withdrawals into a token pool from their private accounts and addresses (which are labelled via S). To call a balance of a token pool at time t (or an individual account) write $\beta_j(t)$ for $j \in J$. In the instance where multiple token types are allowed, write $\beta_{j\theta}(t) = \beta^t_{j\theta}$, where θ references the type of token for which the balance is being called. For example, if there are two tokens in our universe, token-1 and token-2, then $\theta \in \{1, 2\}$ and two pools $J = \{1, 2\}$, then for a fixed $t = 0$, the set of balances are represented by $(\beta^0_{11}, \beta^0_{12})$ for pool-1 and $(\beta^0_{21}, \beta^0_{22})$ for pool-2, over the respective tokens. Dynamically, we are interested in changes to these balance variables over time as deposits and withdrawals are processed through the pools.

2.1.4. *Policies.* To model insurance state space, a list of policies is required. Let L be a list of policy types and suppose $L(t) = L^t$ is the list of policies at time t. This can be thought of as a global list of policies that underwriters will underwrite, and policies holders can buy. A policyholder $h \in H$ holds a policy $\ell \in L$ or some set of policies $L_h \subset L$. The pair (h, ℓ) is called a member-policy pair. The *status* of member-policy pair may either be valid or invalid. Given a member-policy pair, we suppose we can query the status of the pair for any t at a negligible transaction cost: $\text{status}(h, \ell) = \texttt{True}$ or $\text{status}(h, \ell) = \texttt{False}$.

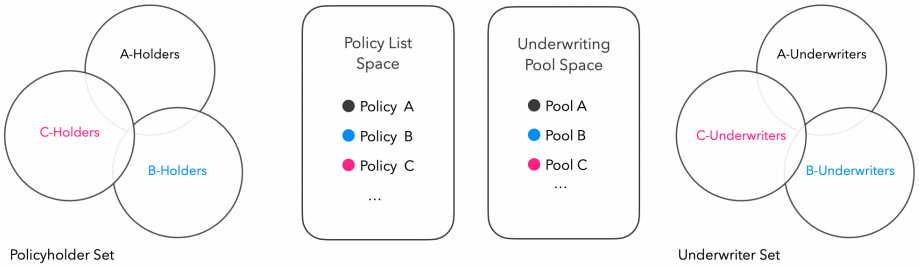

FIGURE 1. Policyholder-Underwriter and Pool-Policy Interactions.

2.1.5. *Premiums.* Member-policy pairs attract premiums. Given a pair (h, ℓ), let $\rho(h, \ell)$ be the premium payable by member h for policy ℓ. Similarly, let $\rho_{h\ell}(t) = \rho^t_{h\ell}$ represent the premium payable by member h for policy ℓ at time t. To get the total premiums paid into the system, take the sum over policyholders, policies and time. This is simply the statement that premiums need three coordinate points to identify them correctly: the time of payment, the policyholder and the policy type.

2.1.6. *Claims.* Claims are intertemporal in that a claim that is lodged during t may only be settled at t'. Every claim may be thought of as a ticket with metadata: an issuer (the policyholder), a policy-type, a unique policy identifier and a creation timestamp; as well as any restrictions on where the claim can be paid from. Let C be the system claim space; which is a record of all claims generated. We suppose that $C^t \subset C$ is a container for all claims generated up to and including the end of round t; and $C^0 \subseteq C^1 \subseteq C^2 \ldots \subseteq C^t$. Conveniently, $C_0 = \emptyset$ and there are no claims at time $t = 0$. Let C^t_ℓ be those claims pertaining to policy $\ell \in L^t$. Claims with respect to different policy types do not overlap, so C^t_ℓ and $C^t_{\ell'}$ are disjoint and can be used to partition the claims space C^t_ℓ.

Every claim $c \in C$ has a primary status: resolved or unresolved, and a secondary status: resolved-true, resolved-false or unresolved-undisputed, unresolved-disputed. Claims which are resolved-true or resolved-false are deemed to have reached finality, while, on the otherhand, claims that are disputed are assigned a dispute counter which tracks the number of escalations that have occurred with respect to the given claim. The number of escalations is finite; so eventually all claims reach finality. We suppose that creating a claim, resolving a claim and escalating a claim incurs a transaction cost that is borne by either the policyholder, the insurer or a combination of the two. Further, with disputed claims at each level of escalation, the cost of decision making on the claim is expected to increase: claims resolution is without doubt, the most expensive aspect of insurance. We suppose that querying the status of a claim is a negligible transaction cost.

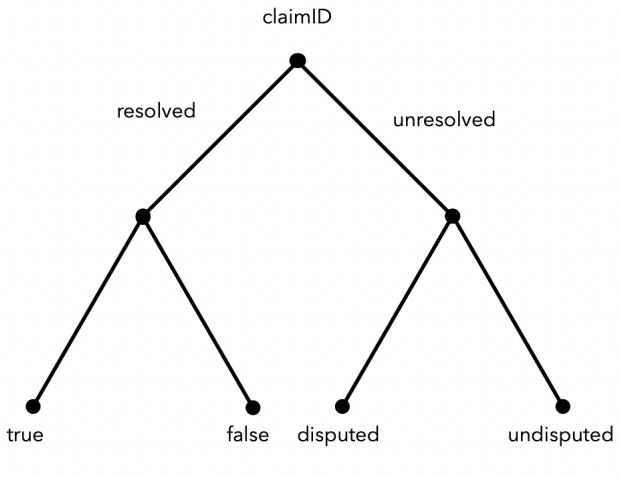

FIGURE 2. Tree diagram of the claims resolution process.

2.2. **Oracles and Authorities.** In the previous section, we alluded to the fact that oracles are necessary in order to query the status of a policy and of a claim. Public ledgers act as trusted data commons which support multi-agent verification of claims; along with data appending functionality (adding blocks to a blockchain, for example) and data agreement models (consensus mechanisms). In proof-of-work mechanisms, the goal of mining is to win the authority over the system's data appending functionality by being first past the post in a computational challenge. Similarly, we need notions of *trusted oracles* and *trusted authorities* to continue our abstract exploration of insurance.

2.2.1. *Oracles.* Let $O \subset S$ be a set of economic agents who are considered to be oracles. A member $o \in O$ returns an output for an input function $f(\cdot)$ and which is valid over some domain. In our version, the querier also specifies the input range of the domain for which the output of the function is required. Generally, a query is of the form $y_o \stackrel{?}{=} f_{st}(D_s)$ where y_o is the data returned by $o \in O$ and f_{st} is the function submitted by $s \in S$ over a specified domain D_s at t. Queries incur a transaction cost which may be paid by the submitter, the oracle, the system or some combination of these.

2.2.2. *Authorities.* Let $A \subset S$ be a set of economic agents called authorities. Authorities have the right to commit data returned by oracles into public record, and add this data to a data commons. If oracles are to be relied on to know private versions of 'truth'; then authorities are to be relied on to make these versions 'public'.

Various systems have already been created to select authorities: Juries, for example, are randomly selected members of the public who collectively form an

authority within a particular legal context. Juries create verdicts, which are public commitments of the truth.

2.3. **Claims Validation.**

2.3.1. *Evaluators.* Let $E \subset S$ be a set of evaluators and $H \subset S$ be a set of policyholders. A policyholder $h \in H$ submits a claim $c \in C$ to an evaluator $e \in E$ for evaluation. The role of an evaluator is (i) to verify that a given policy is valid and (ii) to assign a combination of members, oracles and authorities to evaluate the validity of a claim on the policy. Further, multiple evaluators may deal with a single claim.

2.3.2. *Validator Committees.* Let $c \in C$ be a claim. Any subset of $K \subset S$ with $|K| \geq 2$ is termed a validator committee. Moreover, let $f_j^*(t, c)$ be a function that takes as inputs t, c, and for a given member $k \in K$. Given the outputs $\{f_1^*(t), f_2^*(t), \ldots, f_{|K|}^*(t)\}$ a committee is able to produce an output using this set of inputs – a validator committee is able to collate the outputs of its members into a single output; a decision. More specifically, a validator committee is able to produce an output that resolves the state of a given claim.

2.4. **Dynamics.** The abstractions above are useful for us to describe the general state of an insurance system as a snapshot of policy creators, policyholders, underwriters, claims evaluators, token pools, policies, claims, oracles and authorities.

During a given round t different dynamical behaviours may occur which modify the current state as well as the governance of the system as a whole. These include the following:

- *Add or Remove a Policy Type.* A member that has policy creation or deletion rights, adds a removes a policy from the list of policies.
- *Sell New Policy.* A new member may join the set of policyholders. A policy token may be issued to validate this membership.
- *Receive Payments.* Premiums may be paid into a collection pool by individual members. We may also have separate pools - pools for collecting premiums, and pools for collecting underwriting pledges. Underwriting and premiums may be paid in different token types. Changes to these pools track the exchange of insurance risks between agents.
- *Update Pool Balances.* A token pool may pay out on a set of valid claims; and accumulate deposits from various sources, including additional underwriting capital deposits and premiums. In the situation where the funds are managed, then interest on the pool may be allocated into the pool. Premiums may be collected into the pool. Underwriters may withdraw capital from a pool.

- *Update Pool Parameters.* An additional layer of complexity is that pools may be governed by a subset of the members of S. In this case, the parameters which are used to govern a given pool may be altered; by the authorities that have the right to govern the pool.
- *Update Policy Validity.* Any given policy may be deemed valid or invalid. Policies expire, premiums on policies may or may not be paid and so forth. This means that at each timestep it is necessary to update the validity of different policies. This is to prevent the system from paying out for a invalid policy or accepting a premium payment for an invalid policy. Again, a subset of oracles and authorities can make this decision; note that these entities can be created as a smart contract.
- *Create New Claim.* A member with a valid policy may create a new claim against that policy.
- *Update Claim Validity.* An authority or a validator committee working in conjunction with a set of oracles may update the status of a list of existing claims and move them toward finality.
- *Modify Member Designation.* A given agent $s \in S$ may have multiple designations. An agent might be a policyholder and an underwriter; or an authority in one instance and an oracle in another. During a given timestep, a member may also be added to a validator committee as well as removed from a validator committee. Evaluators can, for example, form a committee that has the job of evaluating a set of claims on a list. These actions are designation modifiers. Finally, in the evolution from S_t to S_{t+1}, new members may join the set S; so there is a change in designation from non-member to member. Note that once a non-member has joined the set S, we suppose that they become dormant as opposed to leaving the set. Thus, the set S^t contains the set $S^{(t-1)}$ and so forth.

2.5. **System State.** Understanding the system state requires keeping track of the evolution of pools, policies, claims, holders and underwriters as well as any fees attracted through operations. We use the general language above to work through specific examples.

Example 1. Let $H = \{a, b, c\}$ be policyholders and $U = \{x, y, z\}$ be underwriters. Suppose there is a single policy type and $L = \{\ell\}$ and two pools $J = \{1, 2\}$ and $t \in \{0, 1, \ldots, T\}$. Let $\theta \in \{1, 2\}$, we track the balances in pool-i as $\vec{j}_i(t) = (\beta^t_{i1}, \beta^t_{i2})$. There are no transfers between pools only underwriter withdrawals and deposits, and claims payments. Let $\vec{x}_i(t), \vec{y}_i(t), \vec{z}_i(t)$ represent the net underwriter behaviour for our respective underwriters in U and pool-i. Let $\vec{a}_i(t), \vec{b}_i(t), c_i(t)$ be

the behaviour of policyholders. The state of pool-i in round $t+1$ is:

$$\vec{j_i}(t+1) = \vec{j_i}(t) + \vec{a_i}(t) + \vec{b_i}(t) + c_i(t) + \vec{x_i}(t) + \vec{y_i}(t) + \vec{z_i}(t).$$

We can simplify this further, by aggregating underwriters and holders. Another simplification is to suppose that each pool handles only one token type. The next step in terms of modelling is statistical, for example, letting p_1 be the average probability that a premium is paid, and p_2 be the average probability that a claim is paid. Let q_1 be the average probability that an underwriter requests a withdrawal, and q_2 be the average probability that an underwriter makes a deposit.

Example 2. Consider the static situation where the number of policyholders is fixed and each holder has the same policy. Suppose N is the number of policyholders and α is the average premium. The expected premiums paid in a given round is $p_1\alpha N$. Next, suppose that a claims payout is on average χ, in which case $p_2\chi N$ is the amount paid in claims. After T rounds $(p_1\alpha - p_2\chi)N \times T$ is the net expected amount paid by policyholders less claims. A pool holds some initial underwriting capacity β^0 and let M be the number of underwriters. The withdrawal and deposit behaviour from underwriters would be given by $(q_1\mu - q_2\nu)M \times T$. The long term balance in the pool is given by the sum:

$$\beta^t = \beta^0 + (q_1\mu - q_2\nu)M \times T + (p_1\alpha - p_2\chi)N \times T.$$

We can simulate this as a turn based game. The game parameter set is:

$$(M, N, q1, q2, p1, p2, \mu, \nu, \alpha, \chi, \beta, T).$$

Figure 3 below shows the outputs for the configuration:

$$(10, 5, 0.7, 0.6, 0.4, 0.5, 20, 30, 10, \chi, 1000, 5000)$$

. The value of χ is set at 50 (upper), 55 (middle) and 60 (lower) respectively in below. Notice the significant difference in the pool balance and the volatility as we increase χ. What this shows is that finding the balance between underwriters and policyholders through pricing, and protecting both parties' interests is the key role of the insurer.

Example 3. Suppose that a group of size N creates an underwriting pool with capacity β and that the probability of any one of them being affected by some event is p. The game lasts only one round, so we can neglect the time element. If they are affected by the event, the payout is δ and if they are not affected, then the receive an equal share of the remaining pool. The payout made is given by $p\delta N$, and we have $\beta - p\delta N$ remaining. Claimants receive δ, and non-claimants

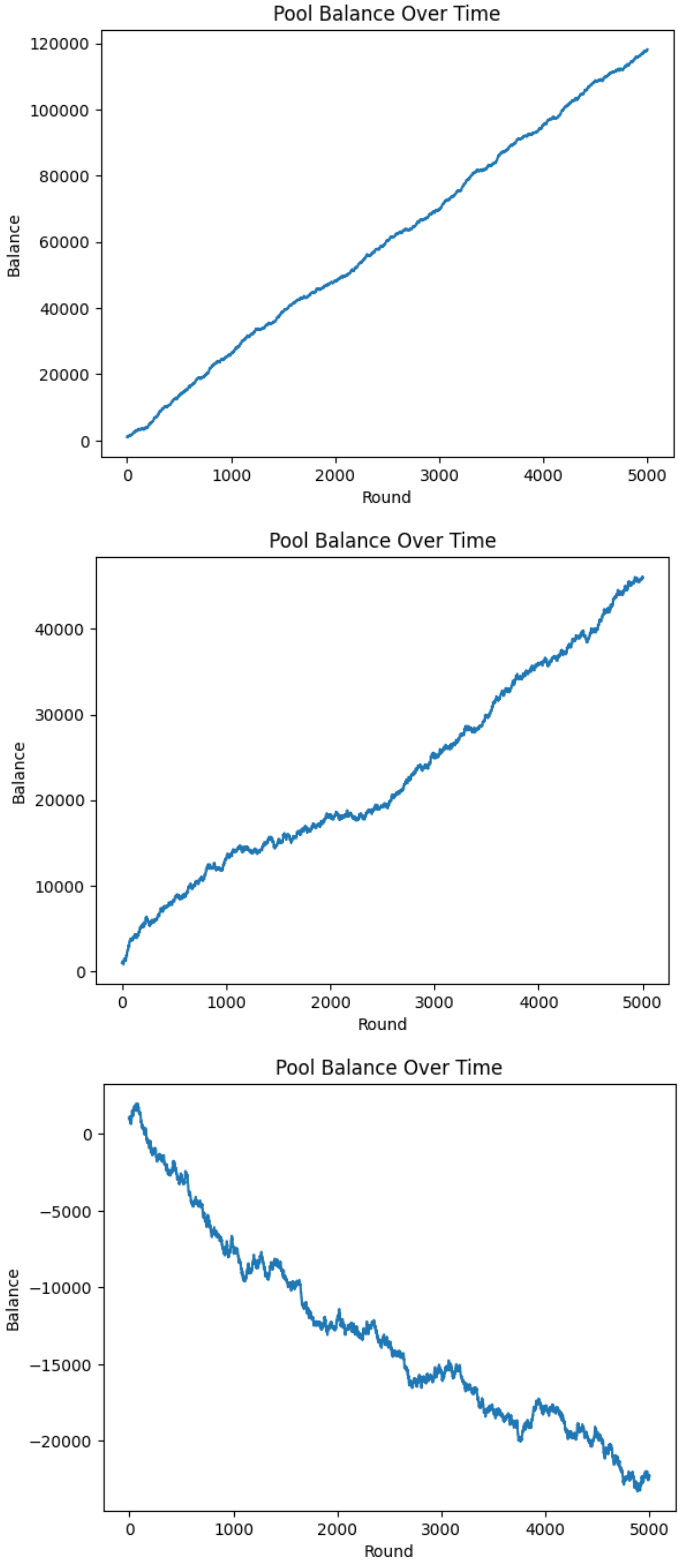

FIGURE 3. This figure shows the above game with different configurations of the parameter χ and all other parameters held constant. We have χ set at 50 (upper), 55 (middle) and 60 (lower).

receive on average $(\beta - p\delta N)/(1-p)N$. The restriction here is that $p\delta N \leq \beta$ or the claimants are not covered by the amount in the pool.

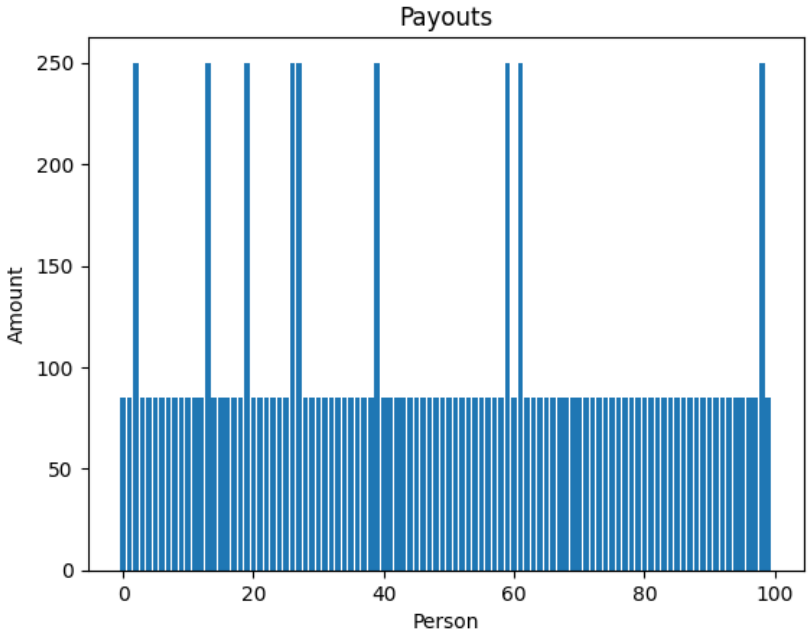

FIGURE 4. This demonstrates the Example 3. where $N = 100$, $\beta = 10,000$ and $\delta = 250$ with $p = 0.05$.

Example 4. Suppose that $S = \{a, b, c\}$ and we have Alice, Bob and Charlie as protagonists. At $t = 0$, each of them pays a random amount between 100 and 1000 into a pool. For each timestep, there is a probability of 0.15 they make a payment of between 100 and 250 into the pool, while there is a probability of 0.1 that they pool gives them a random payout. This creates a type of mutual game which is relatively easy to simulate. Figure 5 shows the behaviour of this system with payout limits of 50 and 500. Generally, there is a catch-up problem between pool withdrawals and pool deposits.

3. The Plural Ecosystem

Insurance systems are a combination of the elements discussed in Section 2: Underwriters and claims validators add value to the system by providing collateral and services. Policyholders add value through the fraction of their premiums that pass into underwriting, while removing value through the claims process. The token acts as a tool for managing the relationship between underwriters, policyholders and claims validators. In addition to these, oracles and data providers form part of the economy as sources of 'truth' to decision makers in the token system, particularly claims validators.

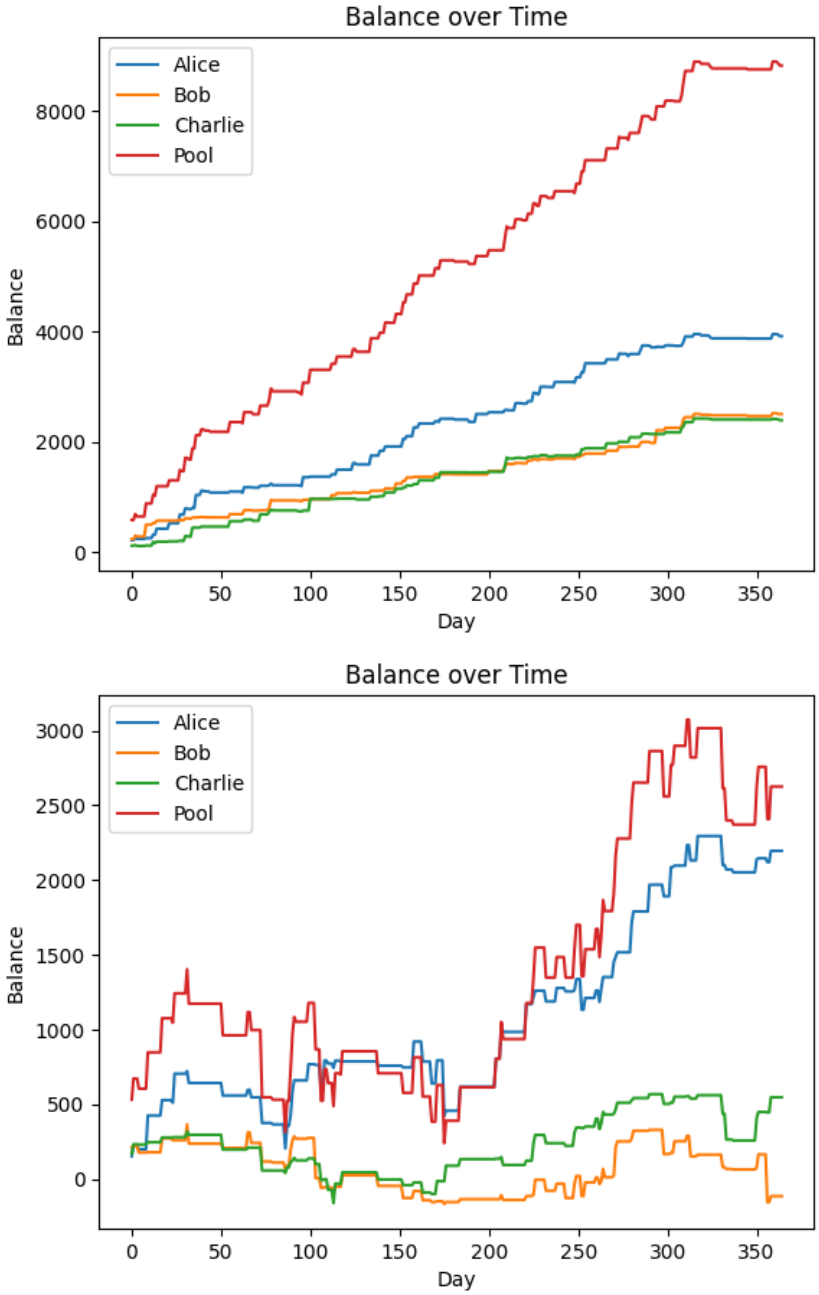

FIGURE 5. The upper graph shows the behaviour of the pool in Example 4. with a payout limit of 50, and the lower graph shows the behaviour of the pool with a payout limit 500. In the latter case, Bob, for example, has overdrawn his overall deposits into the pool.

The Plural ecosystem has two core components: a decentralized autonomous organisation (DAO) and an equity company: respectively Plural DAO and Plural DATA. Plural DAO is responsible for underwriting management and governance, and Plural DATA is responsible for creating insurance products, and researching and developing claims validation technology on behalf of Plural DAO. At

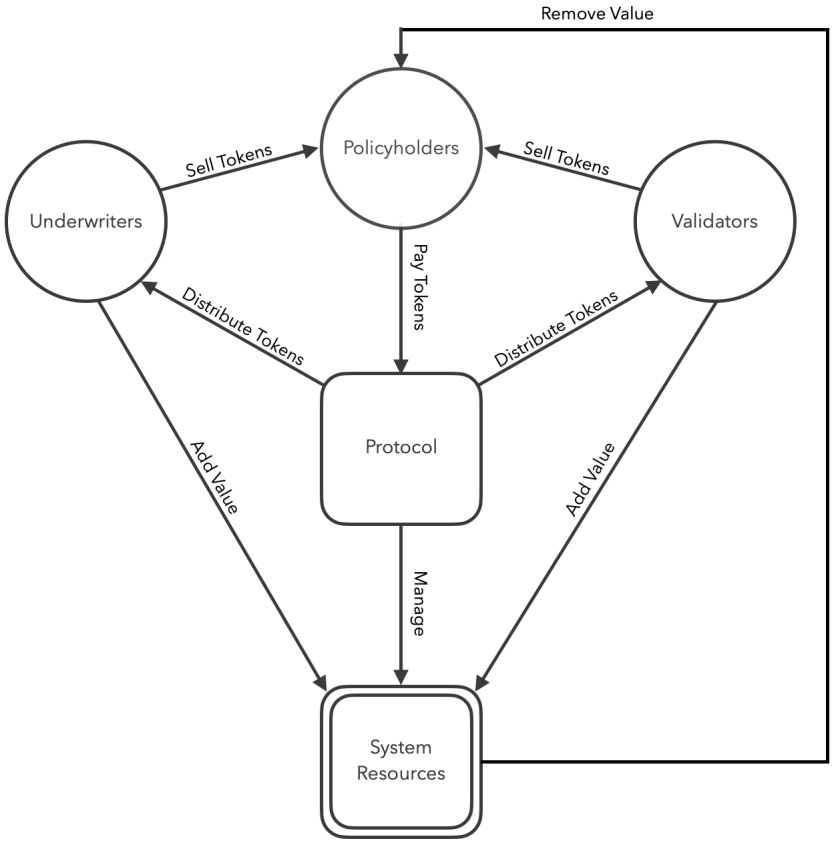

FIGURE 6. Value flows through an insurance token system.

inception, there is a close-symbiotic relationship between these entities in order to support the successful establishment of the ecosystem. The system is setup in a stake-to-validate structure, so Plural DATA is the first validator in the system with a locked stake. Once the ecosystem is established, multiple validators responsible for a spectrum of policy products may be envisioned. In the remainder of this section, we discuss the mechanisms used to implement the ecosystem in practice.

3.1. **Underwriters.** Underwriters may make deposits into a given pool, and may make withdrawals from a given pool. When a deposit is made into the pool a non-fungible token (NFT) is created and sent to the underwriters designated proxy wallet (public address). The NFT holds the metadata record of the deposit transaction including a timestamp record, the amount deposited, the token-type, and the *yield method*.

Yield methods are the incentive structures which act to attract underwriters to the ecosystem: these may be asymmetric and in the form of ecosystem governance tokens, or voting rights, as well as symmetric - yields paid in the base currency that is used for underwriting. Initially, all yields will be in Plural governance tokens.

Further, insurance underwriting capital is generally managed by the insurer for the sake of capital efficiency. In the event that pools belonging to the DAO are managed, with permission from the underwriters, the goal of the DAO is not to profit from underwriter deposits; but rather to pass additional value to underwriters where they can be found in the broader DEFI ecosystem.

Suppose that U is the set of underwriters, and U^t is the set of underwriters that exist in the ecosystem at time t. Let $J_U^t \subset J$ be the set of all pools that the underwriters can interact with. The current balance is $\beta^t = \beta(J_U^t)$. In order for the system to remain compliant, the balance in the pool has to be above the prescribed minimum capital requirement (MCR). If $\epsilon^t = \epsilon^t(J_U^t)$ is the MCR for this particular set of underwriters, then we require $\beta^t \geq \epsilon^t$ for compliance. Moreover, the insurer has to remain compliant while their operations are being carried out. This implies a capital buffer, say $\delta^t = \delta^t(J_U^t)$. Generally, the balance has to maintained at a level $\beta^t \geq \delta^t + \epsilon^t$.

The yield per unit of base currency deposited into an underwriting vault or vaults are used to regulate the capital buffer δ^t. The premise is that underwriters are yield-seeking and will respond to increases in yields when the ecosystem moves to up-regulate the capital buffer and decreases in yields when the ecosystem moves to down-regulate the capital buffer. Note that this can be done on a pool-by-pool basis or on a policy product by policy product basis, as required.

3.2. **Policyholders.** Recall that a policyholder is a member $h \in H \subset S$. Each policyholder is the owner of at least one policy from the list of policies L. At time t, we have H^t, L^t and S^t. Policies are issued as NFTs which contain policy metadata for the insurance contract; that is the contract between the ecosystem and the holder. This metadata includes the policy terms, premium payment and escalation terms, and links to a policy record with further details on the relationship between the ecosystem and the holder. Additional metadata for the policy includes the pools that the policy can draw upon for capital underwriting, and reimbursement terms, for example, if the ecosystem has mispriced risk. Policyholders are identified through their policy NFT; for security reasons, a secondary method of identification may be added for confirmation of a policyholder's right to a policy.

The ecosystem tracks policyholders through issued NFTs. Further, the ecosystem tracks capital exposure through NFT metadata. For any given t, the collection of issued policies represents potential future expenses for the underwriters in the ecosystem through claims, as well as income through yields derived from premium payments. We suppose that $\Phi(H^t)$ computes the global capital-at-risk at any given point in time from policyholders; and $\Omega(H^t)$ computes the premiums payable at

any given point in time from policyholders. These are intended to be publicly available and verifiable numbers available to underwriters and regulators.

3.3. **Oracles.** To support claims validation insurance ecosystems consume data. This data may be provided by policyholders, private third parties, and public data source providers, for example, a public blockchain record or an open data commons. Plural DATA is responsible within the Plural ecosystem for claims validation and will, in turn, engage oracles for data queries. For example, querying (and building) blockchain analytics APIs to understand wallet behaviour. We distinguish between public oracles where other entities know that the oracle is a acting as a data source to Plural DATA, and undisclosed private oracles which provide data anonymously to Plural DATA. In either setting, oracles provide answers to queries; and not the datasets themselves, in this way Plural DATA avoids data custody. Extensive work on the area of data management and using data marketplaces has been done already in Web3 by, for example, Ocean Protocol. The claims records generated by the ecosystem will be placed in a public data commons, for example, IPFS, and accessible by oracles in order to verify previous claims processes and actions.

3.4. **Validators.** Validator committees are some combination of authorities, evaluators and data sources which are formed into a decision making group. Let $h \in H$ be a policyholder and suppose that $e \in E \subset S$ is an evaluator. The policyholder 'Alice' creates a claim by submitting a ticket to the evaluator 'Eve'. Eve checks the ticket to ensure that Alice is a valid policyholder and then creates an instance of the validation method which applies to the policy and the particular claim made by Alice. The details of the policy $\ell \in L$ are included in the claim ticket which is created as an NFT through a smart contract interaction.

The evaluator creates a validation committee when claims are lodged against valid policies. In principle, Alice may submit her claim to multiple randomly selected Eves, which in turn create multiple committees. An aggregation method then applies to obtain a single Boolean (True or False) outcome with regard the claim. In the event that the claim is found to be false, a limited number of appeals may be lodged through a claims dispute process by Alice. The record of claim evaluation and processing is also public and verifiable.

An assumption that we make is that we can validate claims in an adversarial environment where the policyholder may also be implicated in the claim. To the extent that we can use data, and sufficient data is generated to validate a given claim, data will be used in our setup. So, for example, in the case of parametric insurance for farmers, weather data, time lapse aerial photographs, and satellite data may be used to validate a claim and pay the policy holder.

Claims processing through data presents an opportunity for blockchain and, more broadly, distributed ledger systems to create large data marketplaces which can be used for validation and pricing purposes by insurers. Further, this approach to claims validation allows the original data creator to remain the data owner without passing data custody on to the claims validator. It is possible to compute over a dataset and read only the result of the computation and not the data itself. This opens the potential for data re-usability for data creators; and the offsetting of insurance through third party data selling. For example, using your health data to reduce your insurance premium, regardless of whether that data puts you in a 'good' or 'bad' category when it comes to risk.

In the event that no clear decision on the claim can be made in an automated way using data and automated authorities (via AI/ML), a voting procedure in which multiple (human) authorities examine the claim can be utilized. In any case, each claim generates a claims report to be stored as a record on IPFS, which documents the procedure used to evaluate the claim as well as the outcome of the claim.

Claims validation committees may be created from a pool of Plural DAO members in the same way that members of a jury are selected. Further, in order to have this right of membership and benefit from any claims processing incentives, members will need to stake collateral. In the event that claims which are later established to be false are approved, and claims which are shown to be true are not approved; validators will be punished through a loss of collateral.

3.5. **Plural Ecosystem Models.**

3.5.1. *Policyholder-Pool Relationships.* For a fixed t, let X be a set of policyholders and suppose Y is a set of premium collection pools. The relationship between policyholders and pools is a bipartite graph on the set $X \cup Y$. A policyholder $x \in X$ is labelled by their public wallet address and creates a payment transaction to a pool treasury contract $y \in Y$. This is represented by a weighted edge $w(x)$ which records the value of the transaction. To get the total premiums paid by a particular policyholder x into a particular pool y we take the sum $\sum_t w_t(x, y)$.

To get all payments made into the pools is slightly more complex, since we will need to reconcile different token types; and use a base currency to arrive at a single number. To avoid this complexity (and to support value in the ecosystem) we suppose that all policy premiums are paid in Plural governance tokens. With this caveat, total premiums paid by a policyholder is the sum of weights $\sum_{t,y} w_t(x, y)$. This value may also be built up from payments for particular policies $\ell \in L$. We use these variables to understand premium payment behaviour on the underlying

network. For a given t, we are interested in actual payments versus anticipated payments, and so forth.

3.5.2. *Underwriter-Pool Relationships.* For a fixed t, let U be a set of underwriters and suppose that V is a set of underwriting pools. Underwriters may also be policyholders, and this makes sense so as to attract people who would otherwise self-insure into the ecosystem as both underwriter and policyholder. The relationship between underwriters and pools is a bipartite graph on $U \cup V$ and underwriting payments are modelled as weights $w_t(u, v)$.

In principle, underwriting deposits can be made with tokens of different types, so we may consider weights of the form $w_t(u, v, \theta)$ where θ is the token type. The balances in underwriting pools may be *managed* or *unmanaged*. Unmanaged balances are capital inefficient, but available for immediate claims payouts. On the other hand, managed balances may be at risk from 3rd party failures, and the like. Our aim is to build the ecosystem slowly, so initially, the balances in the underwriting pool will be unmanaged; and the only sources of yields for underwriters will be reserve governance tokens held by the Plural DAO and a percentage of tokens which are received from premium payments. The underlying yield model is calibrated toward capital availability and risk. The more available the underlying capital is the higher the yields, since this capital provides immediacy of cover to policy holders. Capital-yield parameters are set at the pool level, rather than at the underwriter level, so that underwriters can distribute their capital pools and calibrate their exposure accordingly.

For simplicity sake, the aim is to start with a small number of pools and products, and then scale up toward more complexity from there; based on policy demand and underwriting supply.

3.6. **Validator-System Relationships.** Initially, the Plural DAO will rely on Plural DATA to build claims verification technology and develop insurance products as well as their corresponding risk models. The role of Plural DATA is to build the necessary infrastructure to enable robust and scalable claims validation and payments. Validators are required to stake ecosystem governance tokens in order to qualify as a validator (either as a private-off chain oracle data source, an evaluator or an authority). This requirement also applies to Plural DATA.

Suppose that $Q \subset S$ is the set of validators; a validator $q \in Q$ has a current stake of $\psi_q(t)$ in a pool $J_q \subset J$. Each validator has their own staking treasury. As validators process claims, validators receive ecosystem tokens into their pool J_q as *validation reward*. The rewards tokens are held in an escrow account until a claims dispute window has expired. If a claim outcome is disputed, the validation reward is delayed until the dispute is resolved. Validators whose claims are overturned on

appeal may also lose a fractional percentage (per overturned claim) of their staked tokens as a punitive measure; these tokens will be either be burned or recycled as a policy reimbursement. Further details on the ecosystem dispute resolution process and protocols will be released at a later stage: The overall goal is to systemically reduce the number of false-positives, and false-negatives (the claims error rate) through a robust incentive structure for validators and a fair appeals process for claimants.

4. Tokenomics

We take the view that tokenomics is about incentive systems and structures for ecosystem participants. This is different from the token breakdown and release schedules which is often thought of within the wider crypto-space as tokenomics. That said, the full token allocation breakdown and release schedule will be made public via our official Gitbook. This section describes the governance token's functions and behaviour as well as the mechanisms via which the governance token is intended to be deployed.

4.1. **Governance.** Plural DAO is a decentralized organisation that is member run. Membership is a function of ecosystem participation and resource commitments. Policyholders and underwriters are automatically members in the DAO. Validators are also members in the DAO. Voting tokens are generated staking mechanisms: members will need to stake tokens in order generate voting tokens. Members are able to vote on policy implementations, changes to the validation structure and architecture and other proposals relevant to the DAO, as well as yield proposals and underwriting collateral management. We expect that governance of the DAO is going to be an ongoing iterative work in progress wherein we balance the needs of agency and autonomy, inclusivity, stakeholder participation and long term viability.

4.2. **Payments.** Policyholders are mandated to pay their premiums in the ecosystem token. This does not preclude using other tokens to pay premiums and this is an option that may be explored in the future. However, initially, to drive the adoption and decentralization of the ecosystem token this requirement is in place. Premiums are collected into payment pools, and then allocated automatically to support the functions of the ecosystem: underwriting and validation, as well as operations. A percentage of collected premiums are held in a long term escrow; in order to build up an ongoing token reserve.

4.3. **Staking.** Ecosystem tokens are staked in order to support voting and validation functions. This has the effect of reducing the circulating token supply;

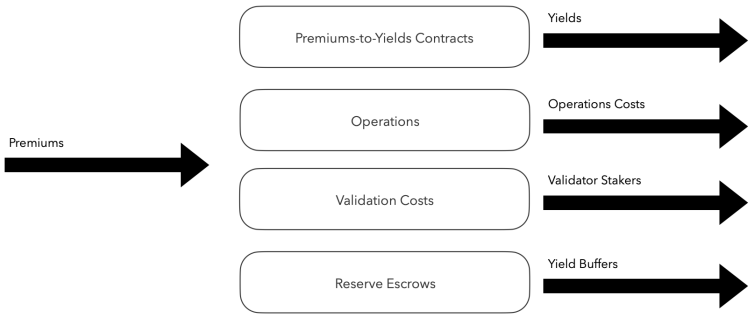

FIGURE 7. Allocation of Premium Payments.

but this is not the purpose or aim of these functions. The goal of the former is governance and the later value-at-risk in exchange for decision making privileges.

4.4. **Yields.** Underwriters receive yields for providing capacity to the ecosystem. These yields are paid in governance tokens based on the amount of capital deposited by the underwriters (measured relative to some based currency) and the availability of capital. In terms of system design, this could be as simple as having a pair of treasuries the first which contains the minimum capital required to cover the existing set of policies, and the second which contains the capital reserve.

There are two sources of tokens which may be used to pay yields: DAO reserves and premiums. In order to bootstrap the initial adoption of the protocol, yields from reserves will be allocated at preferential rates. Yields are paid out on a yield per unit of underwriting capital deposited, with an additional increment that factors in risk. When the protocol is initialized; the capital at risk calculation and dashboard will be made publicly available, as well as the yield per unit risk per unit capital.

A yield controller based on a proportional-integral-derivative (P-I-D) controller system from engineering will be deployed as part of our yield system in order to ensure that there is an efficiency of capital to coverage ratio. As more policies are sold, more capital is required due to an increase in coverage demand. The coverage buffer is the difference between the current minimum capital required for coverage total available capital. We expect this to be similar but different to the controller system that is used in the RAI ecosystem.

As policy demand increases the minimum coverage requirements will increase and use up the the buffer. In turn, this will trigger an increase in short term yields. If there is a drop in policy demand, yields will be reduced and excess capital will move out of the underwriting pools. Essentially, this is the managed system-underwriter-controller feedback loop.

5. Concluding Remarks

The goal of this paper is to act as a conceptual introduction to insurance as an ecosystem, as well as the Plural ecosystem. For the most part, we have introduced a framework for describing various aspects of insurance systems, before describing some of the details around Plural's particular implementation. In this regard the paper is incomplete, and further concepts may be introduced with respect to the structure and behaviour of the ecosystem in the future. In terms of background materials, we have referenced RAI [3] and Ocean Protocol [2] in the paper as well as used concepts from Graph Theory and Set Theory see [5] . For insurance related information on solvency and compliance see [4]. For details on using graph theory and set theory to model payments systems see [1].

It is important to note that creating a token governed decentralized ecosystem is an ongoing process in which ideas, design and feedback interact. Our promise is to iterate on these initial thoughts, validate, test and continue to push the boundaries of the crypto and Web3 insurance space.

6. Acknowledgements

Many people have assisted us in the writing of this document, and we have acknowledged them privately. This acknowledgement is for the staff of Chill #2 Cambridge, 180 Studio Kitchen London, and the Corner Hotel London who, without knowing it, helped us get this paper written. Please reach out to us via Discord or Twitter if you have questions or comments. Further details on the team and project can be found at `www.plural.fi`.

References

[1] V. Naicker. *Token Exchange Games*. `https://arxiv.org/abs/1904.00746`. Accessed: 2023-04-01.

[2] *Ocean Protocol*. `www.oceanprotocol.com`. Accessed: 2023-03-30.

[3] *Reflexer Finance*. `docs.reflexer.finance`. Accessed: 2023-03-30.

[4] *Solvency II: Raising the Bar on Expertise*. `http://www.actuaries.org/CTTEES_TFSP/Documents/Hague_Item3c_GC_Solvency2.pdf`. Accessed: 2023-03-30.

[5] D.B. West. *Introduction to Graph Theory*. Featured Titles for Graph Theory. Prentice Hall, 2001. ISBN: 9780130144003.