

VERITY WHITEPAPER BETA DRAFT



Matt Goldenberg, Doug King, Eric Johnson, Keith Swallow

Table of Contents

ABSTRACT	4
BACKGROUND	4
CROWDSOURCING	4
WISDOM OF THE CROWDS.....	5
INTERNET REPUTATION.....	5
COMBINING REPUTATION AND CROWDSOURCING.....	5
ETHEREUM	5
SUMMARY OF THE VERITY PLATFORM.....	6
REPUTATION MEASURES.....	6
COMMUNITIES	6
CONTESTS	7
AN EXAMPLE	7
REPUTATION	8
REPUTATION AND RATINGS STANDARD	8
VALUES-BASED REPUTATION	9
VALUE TAGS	9
VALUE SCORES.....	9
SKILLS-BASED REPUTATION	13
SKILL TAGS	13
SKILL TOKENS.....	13
VERITY CONTESTS.....	18
CONTRIBUTION-BASED REPUTATION.....	29
UTILITY FUNCTION STANDARD.....	29
CONTRIBUTION TOKENS.....	30
GOVERNANCE.....	30
GOVERNANCE STANDARD	31
ACTIONS	31
PROPOSAL STATES.....	31
PROPOSALS	31
ROLES	32
PERMISSIONS	32
GOVERNANCE SYSTEMS	32

VALUE-BASED HIERARCHY	32
CONTRIBUTION-BASED VOTING	33
SKILL-BASED REPUTARCHY	33
VERITYDAO	34
MAKING A PROFIT	34
REFERRAL INCENTIVES	35
VERITYDAO GOVERNANCE ACTIONS	35
COMMUNITY GOVERNANCE	36
CREATING A COMMUNITY	36
COMMUNITY GOVERNANCE ACTIONS	36
USE CASES	37
DECENTRALIZED APPLICATIONS	37
RATING AND GRADING	38
COLLABORATIVE CROWDSOURCING	38
PORTABLE REPUTATION	38
DECISION MAKING	38
FORECASTING	38
MACHINE LEARNING AUGMENTATION	39
CONCLUSION	39
BIBLIOGRAPHY	39
APPENDIX A - MATHEMATICAL BUILDING BLOCKS	40
UTILITY FUNCTIONS	40
SCORING RULES AND DIVERGENCE MEASURES	41
POOLING ALGORITHM	42
MONTE CARLO SIMULATION	42
APPENDIX B – MATH FORMULAS AND ALGORITHMS	42
B.1 - EIGENTRUST ALGORITHM	42
B.2 – EIGENTRUST++ ALGORITHM	43
B.3 - RELATIVE RANK ALGORITHM	44
B.4 – VALUE RANK ALGORITHM	45
B.5 – AFFINITY SCORE ALGORITHM	45
B.6 – TOKEN TRANSFER ALGORITHM	45
B.7 - TOKEN TRANSFORMATION ALGORITHM	45

B.8 – PSEUDOSPHERICAL SCORING RULE EQUATION	45
B.9 – PSEUDOSPHERICAL INFORMATION GAIN EQUATION	46
B.10 – POWER SCORING RULE EQUATION	46
B.11 – POWER INFORMATION GAIN EQUATION	46
B.12 – CREATIVITY SCORES ALGORITHM	46
B.13 – CLARITY SCORES ALGORITHM	46
B.14 – INFLUENCE LIMITER ALGORITHM	46
B.15 – EXPERT STAKE ALGORITHM	46
B.16 - CLIENT	46
APPENDIX C – TECHNICAL DETAILS AND CHALLENGES	46
EXPENSIVE COMPUTATIONS	46
STATE CHANNELS	46
INTERACTIVE VERIFICATION	47
GAS COSTS	48
SHORT TERM – MARKET SELECTION AND FOUNDATION SUBSIDIES	48
LONG-TERM – GAS AUTOMATION AND WEBSITE SUBSIDIES	48
RANDOMNESS	48

VERITY WHITE PAPER

A decentralized governance and reputation system for building collaborative communities

ABSTRACT

We present a programmatic method to crowdsource expertise and contracting tasks, leading to better outcomes at a lower cost than traditional contractual methods. Using advanced optimization algorithms, this method consistently yields credible reputation metrics that measure an individual's values and skills, and can be further extended to provide decentralized governance structures for organizations - a mechanism we call reputarchy.

Two distinct reputation types are used to rank individuals: values and skills. By using these reputation types in crowdsourced contests, we can optimize results according to an arbitrary utility function and update the individual's reputation accordingly. This reputation can be transferred between communities without diluting its validity, allowing for the creation of a universal reputation network.

Using cryptocurrencies and programmable blockchain smart contracts, this paper presents an expertise and contracting marketplace, to incentivize competition and cooperation using credible reputation metrics. This system called Verity, combined with several distinct contest types, can be used to replace stand-alone reputation schemes within most crowdsourced communities and marketplaces. Such communities are a microcosm of our vision: **enable global collaborative networks, representing every conceivable aspect of human ingenuity and judgement.**

BACKGROUND

CROWDSOURCING

The idea of paying a large group of diverse individuals to solve a problem has been around for hundreds of years. In 1714, the British government offered £20,000 (equivalent to ~\$3,500,000 today) to the first individual that could create a method to determine accurate longitude within half a degree¹. This was one in a long line of instances where governments used crowdsourcing contests to spur creativity. Nearly 50 years later in 1761 the problem was solved in two separate ways using two separate measurement methods.

¹ <https://longitudeprize.org/history>

WISDOM OF THE CROWDS

Another large leap regarding crowdsourcing was made in 1907, by Francis Galton. He realized that by taking the median guess of an ox's weight among a diverse crowd of guessers, he could arrive at a number very close to the true value of the ox's weight². This marked the beginning of the realization that crowdsourcing could be used not just for creativity, but for accuracy as well. This idea came to be known as the wisdom of the crowd. Today, the concept of crowdsourcing and wisdom of the crowd is used to power some of the most popular websites on the internet, including Wikipedia, Reddit, and StackOverflow.

INTERNET REPUTATION

In combination with the rise in crowdsourcing, there has been a surge of interest in internet reputation metrics. One early example was eBay's reputation system, which combined positive and negative feedback. eBay moved to show percent negative feedback in 2003³, a tacit recognition that their reputation system was having unintended consequences. Similar realizations ultimately led to the development of several new reputation algorithms designed to resist manipulation, with one of the most popular, EigenTrust, being cited over 4,000 times⁴.

COMBINING REPUTATION AND CROWDSOURCING

As reputation and crowdsourcing began to evolve in tandem, game theorists began wondering how they could be used together to elicit better outcomes from the crowd. In 2011, the United States government agency IARPA created the Aggregative Contingent Estimation project (ACE) to answer such questions⁵. One particular participant in ACE, the Good Judgement Project, was able to achieve spectacular results, using civilians to outperform CIA analysts by 30%⁶.

ETHEREUM

In late 2015, Ethereum launched, a platform for creating smart contracts on the blockchain. For the first time, decentralized applications (dapp) became almost as easy to build as standard web applications, and money was as almost as easy to program as any other piece of data. This represented a way to measure and store metrics such as expertise and reputation that are immutably linked to identities, a feature previously found only in real life reputation. The programmable money aspect also provided intriguing possibilities for incentivized crowdsourcing.

² Galton, F. (1907). "Vox populi". Nature, 75, pp. 450–451.

³ <http://presnick.people.si.umich.edu/papers/postcards/PostcardsFinalPrePub.pdf>

⁴

https://scholar.google.com/scholar?ion=1&espv=2&bav=on.2,or.r_cp.&biw=1366&bih=667&dpr=1&um=1&ie=UTF-8&lr&cites=4085997296011241419

⁵ <https://www.iarpa.gov/index.php/research-programs/ace>

⁶ <http://www.npr.org/sections/parallels/2014/04/02/297839429/-so-you-think-youre-smarter-than-a-cia-agent>

While Ethereum represents a huge paradigm shift in the power to create applications that don't have a centralized failure point, many have pointed out that there are still fundamental limitations to the type of decentralization that can be achieved⁷. In particular, critics have noted that any connection to external data feeds, and any logic that requires human judgement, cannot be decentralized purely with smart contracts. Critics theorize that data must be decentralized, or self-generated inside the decentralized application itself. In addition to crowdsourcing and reputation uses, Verity precisely fills this hole for general decentralized applications.

SUMMARY OF THE VERITY PLATFORM

REPUTATION MEASURES

Verity is a platform that aims to connect communities of experts on the internet into a global network with quantifiable reputation scores for both the values they exhibit, the skills they possess, and the contributions they make to their community. It creates a general reputation standard, upon which it builds its key metrics.

Values are ranked through a web-of-trust system, and are assumed to be transitive meaning those whom exhibit values are assumed to be better at judging those same values in others. Skills are ranked by a system of competition where experts who perform well earn reputation from those who perform poorly. In this way skills based-reputation measures the relative skill-set of an expert compared to other experts. Finally, contributions are ranked by measuring every expert's contribution against a pre-defined community goal. In this way, contribution scores are a measure of how much an expert has helped a specific community.

COMMUNITIES

A **community** in Verity is a group that share commonalities, work together, and share common rules and reputation scoring metrics. This set of shared rules and measurements cause communities self-organize around economic and social incentives.

Checks and balances in the Verity protocol encourage community collaboration. Community core values and goals can be explicitly defined, but norms for what those values and goals mean are flexible and moderated by our value score algorithm. Community members share in rewards and common good to the extent that they are recognized by their peers through reputation and can be rewarded percentages of income based on outcomes derived from contests and reputation score.

Verity is unique in its ability for experts to transfer between communities while maintaining their reputation. This ability is what allows Verity to be a global measure of reputation, instead of a series of one off metrics such as Reddit karma or STEEM power.

⁷ <http://www.truthcoin.info/blog/contracts-oracles-sidechains/>

CONTESTS

Contests are at the center of the Verity protocol. Contests in Verity describe repeated games, through which members can gain or lose reputations, and through which various options can be ranked against each-other. In practice, a contest creator (**client**) is seeking a good (service or product) to be provided at the maximum level of quality and minimum of cost possible in the marketplace. A contest can be seen as a task to be fulfilled or challenge for experts to solve. The client will define contests which describes what type of results the client desires, the type of work involved and the bounty awarded for the winners. The tasks are then offered to experts, who are economically incentivized to work as a team to generate the best submissions. Teams utilize experts to analyze, inform, or rate creative output of value to the client.

While this crowdsourced work can be traditional knowledge work such as creating a website design, contests can also be for things not traditionally considered crowdsourcing. Examples include incentivizing writing social media posts that get upvoted or comments that are considered insightful.

At the end of the decision making process, the contest reward is split between all experts who helped complete the requirements of the contest, with more value going to those experts who performed better.

AN EXAMPLE

As a concrete example of how Verity could be used, let's take the problem of finding experts who can write and audit smart contract code. The hack of The DAO exposed this as an unsolved problem in the Ethereum ecosystem. Using Verity, one would create a "smart contract security" community. This community's reputation metrics and contests would then serve as the backend for a number of important decentralized applications (dapps) needed in the ecosystem such as:

- A dapp for hiring qualified people to code your custom smart contract.
- A dapp for paid, crowdsourced smart contract security audits from qualified professionals
- A dapp for decentralized reputation-based commit access to open source smart contract projects
- A dapp to get advice from other smart contract coders on best practices (with the best advice from the most knowledgeable coders being ranked the highest)

Because each of these dapps would share the same reputation metrics and crowdsourcing tools, each one would in turn improve the accuracy and utility of the other dapps, and the ecosystem as a whole would benefit from new dapps built on the platform in a way not seen with siloed reputation metrics.

REPUTATION

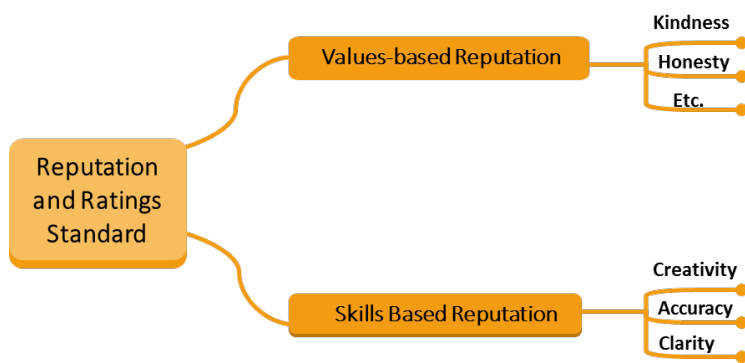


Figure 1 - Verity's Reputation Metrics

Verity's base reputation layer is a general purpose reputation and rating standard, which is a simple standard interface for accessing and understanding reputation and ratings through smart contracts, similar to the ERC20 token standard⁸. On top of this base layer we introduce three reputation metrics that represent the three types of reputation typically used in real life scenarios. One is for **values-**

based reputation, in which individual entities are being rated directly on how other people rate them on community values like honesty or kindness. The second is for **skills-based reputation**, in which an individual skills are ranked based on the output of their skills such as the ability to write forum posts or debug code. Finally, we measure **contribution-based** reputation, which measures an individual's contributions to a community or organization.

REPUTATION AND RATINGS STANDARD

Verity's **reputation and ratings standard** is a set of common functions that allow smart contracts to interact with ratings and reputation in a consistent way, thus allowing them to keep the same interface for their smart contract even if they change reputation metrics.

The standard is simple, and is built on top of Verity's notion of communities. It gives several functions to return information about the reputation or rating metric such as what it should be called, the type of reputation it is (categorical or numerical), and how it should be displayed (limits, units, etc.).

Any community can opt-in to having their members tracked using that reputation metric, and a standard function will return the value of a member's reputation within that community. Finally, if the reputation or rating involves compiling ratings from users, it allows users to assign ratings to each-other and content in consistent ways as well.

On top of this flexible system is built Verity's foundational reputation metrics: its values-based and skills-based reputation metric.

⁸ <https://github.com/ethereum/EIPs/issues/20>

VALUES-BASED REPUTATION

Existing ratings and reputation systems such as Reddit are able to classify the type of content that the community likes, but are not able to identify the type of people that should hold influence in their community according to community values. This means that any open community that uses such a content centric system will be driven by populist values and conflicts.

Verity's values-based reputation is a fundamental type of reputation that records and maps human values in a trust graph. Using values-based reputation, communities can choose **core values** that they want their members to hold. Core values influence things such as ratings or stake in community decisions. We calculate a community 'affinity' score and create incentives for individuals to gain influence in the community in alignment with the values of that community. Values-based reputation eliminates trolling and provocative behavior and can enable application of moral decision-making, such as deciding what content should not be allowed on a platform.

VALUE TAGS

Value tags are simple text representations of the value that will be used in a values-based reputation rating. Each tag will be registered in a global registry along with a URI to a longer free-form description of the value. Long form descriptions can be fuzzy and general, or precise enough to be used in a contractual agreement. For the purpose of Verity reputation calculation, tags and descriptions are arbitrary and only have meaning in the context of the reputation trust graph of the community. The Verity algorithms will work with any language, jargon, or concept.

Tags and descriptions can be created and registered at will, or existing tags in the global registry can be reused if they are a good fit for a community. While users can rate individuals and give personalized local scores to any user for any value, global value scores are calculated only for values which are rated as a core value by some community.

VALUE SCORES

A user can have a **value score** for every value tag that exists on the Verity platform. They are a permanent record for the account that earns them, and hold a value between 0 and 1. Value scores above .5 can generally be seen as having that value, while value scores below .5 can be seen as the reverse. Intermediate values can be seen as degrees of that value that are had.

HOW VALUE SCORES WORK

Value scores represent specific qualities that users of the system hold. They're computed by combining versions of **Relative Rank**⁹ and **EigenTrust++**¹⁰, two Sybil-attack resistant versions of the **EigenTrust** algorithm that normalizes based on the number of ratings each node has been given, and incorporate the structure of feedback that users get.

⁹ <http://dl.ifip.org/db/conf/ifiptm/ifiptm2007/Traupman07.pdf>

¹⁰ <http://www.cc.gatech.edu/~lingliu/papers/2012/XinxinFan-EigenTrust++.pdf>

EIGENTRUST

The EigenTrust algorithm is based on the notion of transitive trust: A peer will trust nodes trusted by those nodes it trusts (and so on). The EigenTrust calculates a local trust value by taking all positive interactions, subtracting all negative interactions, and then propagating this trust transitively along all nodes.

The authors note that for networks with a sufficiently large set of nodes, the aggregated local trust vector t always converges to the same number, regardless of which node i it's calculated from. This means that the trust vector t represents a global notion of trust that the network places on any given node.

START SETS

The authors also note that there are a few problems with the simple algorithm above. Firstly, malicious nodes can create networks specifically designed to increase the score count between each other, by “trapping” the probabilistic crawler mentioned above in a web of trust links.

Secondly, if a node has not rated any peers (or has rated all peers negatively) c_{ij} will be undefined, making the algorithm impossible to compute.

They solve both these problems with the notion of a trusted “**start set**.” This start set represents a set of trustworthy peers that have not been compromised. To remove the chance of the algorithm being trapped in a Sybil compromised network, they make sure that every peer has at least some small amount of trust allocated towards the start set, such that on any given step the algorithm can exit the malicious network by returning to the start set.

If a node has not rated any peers in a positive way, then that peer is treated as having implicitly given its trust to the start set, thus avoiding dead ends in the network. For our purposes, we distribute trust evenly among every node in the start set.

The algorithm for calculating EigenTrust is shown in **B.1 - EIGENTRUST**.

EIGENTRUST++

EigenTrust++ suggests three ways to increase the attack resilience above classic EigenTrust. Firstly, it adds the concept of “feedback similarity” to its trust propagation neutralizing a class of attacks that works by acting honestly, while rating dishonestly (in order to increase your reputation relative to peers who both rate and act honestly). Secondly, it incorporates information about how many feedbacks a peer has received, allowing for peers with lots of negative feedback to be treated differently than peers with low amounts of feedback. Thirdly, it creates thresholds for trust propagation, reducing the possibility that dishonest collectives can pass on reputation to honest nodes, and vice versa.

For the purposes of this paper, we utilize EigenTrust++’s feedback similarity rating, and linear threshold, but don’t use its incorporation of feedback number. This is because we include a different normalization procedure based on feedback number, described below in the section on Relative Rank.

FEEDBACK SIMILARITY

EigenTrust++ notes that a peer can maliciously attack the network by always acting honestly when interacting with high reputation peers, but interacting dishonestly in other situations. It solves this problem by creating a “feedback similarity” metric which allows honest nodes to detect this type of behavior, and propagate less trust to nodes that engage in it. It has an added bonus for our algorithm, as it captures the notion of subjective values – a node will trust other nodes that see the value in the same way that it does.

TRUST PROPOGATION

While EigenTrust counts both feedback similarity as well as trust level equally for the score itself, it recommends that you weight feedback similarity higher when determining trust propagation. It uses some percentage it recommends some percentage β that you should weight the similarity in proportion to the propagated trust score.

The authors of the EigenTrust paper suggest that β should equal .85.

PROPOGATION THRESHOLDS

EigenTrust++ also recommends propagation thresholds, after which trust won’t be propagated at all to a new node. While EigenTrust’s start sets do minimize the damage of malicious collectives, propagation thresholds aim to go a step further and actually penalize these collectives and other bad actors. EigenTrust++ changes this threshold in a random way, to prevent [todo]

The algorithm for EigenTrust++ is given in **B.2 – EIGENTRUST++ ALGORITHM**.

RELATIVE RANK

Relative Rank is an algorithm that seeks to add additional Sybil-resistance to the EigenTrust algorithm, while at the same time making it more suitable for peer-to-peer markets. By transforming EigenTrust’s arbitrarily high trust vectors into a normalized value, Relative Rank creates a clear decision procedure to determine if a peer should be trusted or not within an interaction. The normalization procedure also seeks to include negative feedback, in order to separate dishonest users from users whom have simply not been ranked. In order to create this procedure, Traupman first analyzed the behavior of EigenTrust in marketplaces, then tried to determine a clear threshold in the determination of whether a node was trustworthy or untrustworthy. We’ll use the same procedure but apply it to EigenTrust++.

VALUE RANK

Value rank, our algorithm for calculating the values that an individual holds, makes two minor changes to the original relative rank algorithm. It makes local ratings more granular, and it allows for multiple start sets.

GRANULAR LOCAL RATINGS

In the original relative rank algorithm, feedback is binary, and interactions could be rated only positive (plus one) or negative (minus one). While this makes sense for the original implementation of EigenTrust, in which a peer either gave the correct data or did not, it does not allow for the nuance that comes with arbitrary

values, such as deciding the level of kindness that a user showed. In relative rank, this also means that there is less distinction between individual ratings, because of the high correlation between r and k . For this reason, individual ratings are given as a decimal value between positive one and negative one, allowing for more granularity in every interaction. By multiplying the final relative rank by 2 and subtracting 1, individuals can also get a clear intuition for what a relative rank score means - an individual rating of -0.3 means the same thing as a relative rank score of -0.3 .

MULTIPLE START SETS

In the original relative rank, a single start set is used. A related algorithm is also given called RAW which allows for personalized start sets. In value rank, multiple start sets can be created, one for every community that ranks that value as their core value. This captures the notion of values as relative interpretations that exist within community context, while still striking a good balance with computational cost.

COMMUNITY LEVEL VALUE SCORES

The Relative Rank algorithm used to calculate value scores has the ability to calculate different value scores based on what group of initial "trusted users" it calculates from. This is ideal for growing out a global trust graph while allowing individuals to build value scores faster in the community they're a part of. The ultimate goal for value-based reputation in Verity is to build a global trust graph that spans a diverse network of communities and scales to planetary levels.

COMMUNITY CORE VALUES

In Verity, each community can choose to create its own set of **core values**, and for each of those core values can define one or more community members they consider paragons of that value. This becomes the initial start set from which the trust graph grows. As new members on-board and participate in Novice Matches they are rated by the existing members thereby populating new nodes in the trust graph.

AFFINITY SCORE

As individuals begin to get rated in a community's core values, we begin to get a sense of their affinity for the community. We can mathematically represent this affinity by averaging their score along all core values.

We'll refer to this affinity score for user u in community k as $\text{affinity}(u,k)$

We call this average of all community core values for a specific user that user's **affinity score**. The affinity score is used throughout the Verity platform to make sure that individuals who have influence in the community are in alignment with the values of that community.

VALUE SCORES IN ACTION

Let's imagine how value scores might be used in our smart contract security community.

Firstly, the community as a whole would choose a set of core values. For each set of core values the community would choose a start set of people they considered paragons of that value. Let's assume for examples sake that the core values the community chose were honesty, integrity, and thoroughness. These

core values would then ensure that new members trying to become smart contract experts had to have an affinity with those values, through use of the affinity score. For users who were already on the Verity platform, their influence on the governance of the community would be limited based on their affinity score.

Secondly, value scores could be used in our crowdsourced commit access Dapp to make sure that those who participated matched the values of that open source project. For instance, if an open source project valued compromise, they could limit the influence of any given expert's ability to push through a commit based on how high that expert's "willingness to compromise" value score was.

Finally, value score could be used in a subjective way in our smart contract coder hiring dapp to choose candidates that would mesh with a given project's value system. One can imagine first filtering by candidates who share the hirer's values, and then sorting by skills-based reputation tokens to find the most talented candidate within that group.

SKILLS-BASED REPUTATION

Skills-based reputation is the second foundational type of reputation. While values-based reputation answers the question "What type of person are you?" skills-based reputation answers the question "What are you good at?"

Existing methods of measuring skills typically fall into one of three categories:

- Standardized assessments that try to measure different aspects of an individuals' performance
- A record of the education a person has received
- A fuzzy assessment based on an expert's accomplishments or portfolio.

In Verity, skill measurements are based on an expert's ability to use those skills to fulfill the goals of the people they're working for, relative to other experts. While this definition may still seem fuzzy, Verity's power lies in its ability to define these goals in a precise mathematical way, and to keep an immutable history of an experts performance on these goals that can't be tampered with. These innovations allow Verity to measure skills in a precise way that people looking to work with those experts actually care about.

SKILL TAGS

Skill tags are simple text representations of the different skills that experts can hold, along with a URI to a long form description of the skill. Skill tags can be general and fuzzy, or specialized and precise. Skill tags can be created at will, or existing skills can be reused if they fit the task at hand.

While any user can claim to have a skill, skill ratings only exist in the context of a community. Communities can also choose to use the skill tokens from another community to measure that skill in their own community, allowing multiple website to share the same reputation measure.

SKILL TOKENS

Skills in Verity are measured by the amount of **skill tokens** that an individual expert holds. These tokens are typically "colored tokens" that represent specific skills in communities.

There is also a special type of token called Verity clear tokens that don't yet represent a specific skill in a specific community. This type of token is the only type of token that can be used to create new skills within a community. The following four types of skill tokens exist, which correspond to different skills related to a specific skill tag.

1. You can actually use the skill to solve problems. This will earn you **creativity tokens**.
2. You can critique experts' use of the skill, this will earn you **clarity tokens**.
3. You can grade experts' performance against some standard, this will earn you **accuracy tokens**.
4. You can support the platform through buying tokens in an initial sale of them, or referring active users. This will earn you **Verity clear tokens**.

	Creativity Tokens	Clarity Tokens	Accuracy Tokens	Crystal Clear Tokens
				
How to Earn	Earned for creating submissions that do well in the contest.	Earned for giving explanations that help evaluators make better decisions.	Earned for evaluating submissions accurately.	Bought on open market or earned for referring active users.
How to Use	Can be used to show submissions higher when entering a contest.	Can be used to have more chance of showing explanations when entering a contest.	Can be used to weight evaluations higher when entering a contest.	Can be used to create new communities, or converted to any other token type.

Figure 2 - Verity Token Types

CREATIVITY TOKENS

To earn creativity tokens one must generate new submissions to the contest. If those submissions perform well according to the rules of the contest, one will earn more creativity tokens. Submissions don't have to be in the form of text. An idea could just as easily be presented in the form of code, blueprints, or mockups - allowing for creativity tokens to be used for arbitrary crowdsourcing applications.

ACCURACY TOKENS

To earn accuracy tokens, one must evaluate how well submissions meet the variables that go into the utility function specified by the client. Evaluators provide probability estimates among all these variables for the options generated by creatives. This allows the client to make the decision-theoretically optimal decision.

CLARITY TOKENS

To earn clarity tokens, one must explain the pros and cons of different submissions. You can think of these pros and cons as analogous to comments on traditional crowdsourcing sites. However, these explanations do not just have to be in text, and can be in arbitrary media depending on the application. While the client gets

to see all explanations by every critic, the explanations are only shown to evaluators on a probabilistic basis to enable linear regression analysis of the explanation's impact. This allows clarity tokens to get redistributed to those who most help the predictors make accurate predictions.

ACCURACY AND CLARITY TOKENS AS FORECASTING

When the skills of grading and critiquing are used before the outcome of an event is known (example: a battle plan is critiqued and graded before the battle), they correspond to the ability to both understand and predict the future.

This type of crowdsourcing-based forecasting combined with reputation has been shown to outcompete established experts like CIA analysts by 30%¹¹, and is comparable with the accuracy of prediction markets when the proper algorithms are used¹².

It is this ranking of forecasting ability that gives these meta-skill reputation tokens (clarity tokens and accuracy tokens) equal value to the primary skill-based token creativity tokens.

VERITY CLEAR TOKENS

Verity Clear tokens are a Verity Tokens that have not yet been converted to tokens marked for a particular skill-tag, community, and token type(Colored Verity). Verity Clear can be sold as assets for purposes of fundraising for the platform or given out as rewards for referring active users to the platform. Verity Clear tokens are special in that they are not used as a measure of reputation in themselves. Verity Clear can be transformed by any individual at any time, into any of the tokens above, with any skill tag, in any community-subject to that individual's expected token score for that token type. They are also the only tokens that can be used to originally create a new measure for a skill tag within a community.

TRANSFORMING AND TRANSFERRING TOKENS

One of the current problems with internet based reputation is that accounts can be easily sold if not tied to strong identities, thus making it impossible to know if reputation was earned or simply purchased.

Verity solves this problem by allowing the tokens used for reputation to be transformed into sellable tokens, but distinguishing between tokens that are factored into reputation (activated tokens) and tokens which are not factored into reputation (deactivated tokens). The way it does this is by incentivizing the sale of or transformation of reputation tokens only to people who deserve those tokens, and deactivating the tokens if sold to others. By doing this, it gives people honest ways to profit from reputation tokens they no longer have a use for.

¹¹ <http://www.npr.org/sections/parallels/2014/04/02/297839429/-so-you-think-youre-smarter-than-a-cia-agent>

¹² http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2660628

SIMILARITY SCORE

In Verity, we measure similarity by assuming that communities are similar if people that have more of a token in community a also have more of a token in community b . This relationship is easily calculated using an asymmetric measurement of Pearson Correlation, which is described below. The following example assumes that you are trying to move coins from community a to community b

1. Take the set of all users who's tokens t in community b are greater than 0.
2. For each of these users, create a set of pairs (t_a, t_b) for the tokens they hold in communities a and b .
3. Calculate the Pearson Correlation Coefficient for all n pairs

$$r(a, b) = \frac{n \sum_{i=1}^n (t_{na} \cdot t_{nb}) - (\sum t_{na})(\sum t_{nb})}{\sqrt{(n \sum (t_{na})^2 - (\sum t_{na})^2)(n \sum (t_{nb})^2 - (\sum t_{nb})^2)}}$$

4. For all r below 0, set the similarity score to 0.

$$\text{simscore}(a, b) = \begin{cases} r(a, b) & \text{if } r(a, b) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Note that this similarity score is asymmetric because $t > 0$ for community b , but $t \geq 0$ for community a . This has the effect of correcting the imbalanced similarity rate of large communities, which likely will have high many high similarity rates simply due to chance.

EXCHANGE RATE

In a typical currency, the exchange rate is determined by relative demand for two different currencies.

Verity, being a reputation token, has an entirely different notion of exchange rate that mimics how reputation works in real life. Verity creates a simple asymmetric metric that shows how “similar” two different skills token types are, and then uses this equation to create an exchange rate when converting from one to the other. This is done using linear regression, as described below, assuming the user is exchanging tokens from community j to community k .

1. For all people that hold more than 0 tokens in community j or k , create the pair (t_j, t_k) where each t is the amount of tokens those people have
2. Use linear regression to calculate the line of best fit for all n pairs. Plug in the amount of tokens t to get the naïve rate.

$$\text{naiverate}(j, k, t) = \frac{n \sum_{i=1}^n (t_{ji} \cdot t_{ki}) - \sum t_j \cdot \sum t_k}{\sqrt{n(\sum t_j^2) - (\sum t_j)^2} \cdot \sqrt{n(\sum t_k^2) - (\sum t_k)^2}} \cdot (t) + \frac{\sum t_k - \frac{n \sum_{i=1}^n (t_{ji} \cdot t_{ki}) - \sum t_j \cdot \sum t_k}{\sqrt{n(\sum t_j^2) - (\sum t_j)^2} \cdot \sqrt{n(\sum t_k^2) - (\sum t_k)^2}} \cdot (\sum t_j)}{n}$$

3. Multiply the naïve rate by the similarity score, which lowers the output of the linear regression when it applies less:

$$\text{exchrte}(j, k, t, u) = \text{simscore}(j, k) \cdot \text{naiverate}(j, k, t)$$

TOKEN DEACTIVATION

Token deactivation serves as a mechanism to make it hard for people to use their tokens to claim reputation in an expertise they don't have. Inactive tokens cannot be used in contests. The only way for tokens to get reactivated is through earning the same type of tokens in contests. For every token of the same type earned, an inactive token is reactivated. The flipside is that for every token of the same type lost within a contest, you lose a corresponding deactivated token. These lost deactivated tokens are then given back to the community to be distributed through novice matches. What this means is that owning deactivated tokens causes you to be able to both win and lose tokens twice as fast. This encourages users to only be willing to hold deactivated tokens that they know they have expertise in.

TRANSFERRING WITH EXPECTED TOKENS

When calculating how many tokens should be deactivated during the sale or transfer of a token, the user's entire token portfolio, along with the exchange rate, is used to calculate an "expected tokens" value for the community they're moving into. Every token a user holds in all communities (except the community for which tokens are being bought) is multiplied by its percent similarity with the community token being bought. These numbers are then averaged together, and weighted by affinity to the target community, to calculate the 'expected tokens' that the user should have in that community. The expected tokens e are computed for all n communities the user u belongs to with the following equation:

$$\text{exptok}(u, c) = \text{affinity}(u, c) \cdot \frac{\sum_{i=1}^n \text{exchrte}(j_{iu}, c) + t_{iu}}{n}$$

Where s equals the similarity score between community j and the desired community c and t equals the number of tokens in community i .

Any tokens purchased that exceed this number are deactivated.

This discourages whales in a community from buying or transforming tokens to cement their advantage, because they are likely already above their allotment of expected tokens. It also discourages non-experts from buying influence in contests, because their expected tokens are likely very low. What it encourages is experts from similar communities who don't want to sacrifice tokens through transformation to buy tokens from communities they should do well in. This transfer of crypto-reputation to people who deserve that reputation in real life is the exact behavior we'd like to encourage.

TRANSFORMING TOKENS USING SIMILARITY SCORE

When transforming tokens from one community to another (or one contest type to another), a slightly different procedure is used. The similarity between the two communities is used to figure out how many tokens are deactivated *in each transfer*.

This procedure holds up to the point at which the user reaches their expected tokens for a community. After that point, 100% of transformed tokens are deactivated.

$$transf(j, k, t) = \begin{cases} simscore(j, k) \cdot t & simscore(j, k) > 0 \text{ and } exptok(j, k, t) < t_k \\ 0 & otherwise \end{cases}$$

This procedure preserves the incentives of transferred tokens, and adds the additional incentive to transform tokens into tokens which represent similar skills. This allows skill tokens to take on a price related the value of that skill in the marketplace, instead of being merely an average of every skill in the market.

EXPECTED TOKENS IN ACTION

[todo]

NORMALIZING SCORES AFTER TRANSFER AND TRANSFORMATION

When coins are transferred, all scores are normalized using the equivalent of a fake contest that involves spreading the coins among all the participants, then having the transferring participant's score being raised just enough to win an equal amount of coins from every other participant. Because of the use of weighted averages in almost all cases, this is a relatively simple procedure that only has to be recalculated when users are entering contests. In the case of Bayesian scoring, the entire calibration curve will simply be shifted the appropriate amount, thus not changing the shape of the curve itself.

VERITY CONTESTS

In Verity, the only way to earn skills-based reputation tokens is by entering into a contest. A contest is a community level event in which you're competing against all others experts in the community to create the

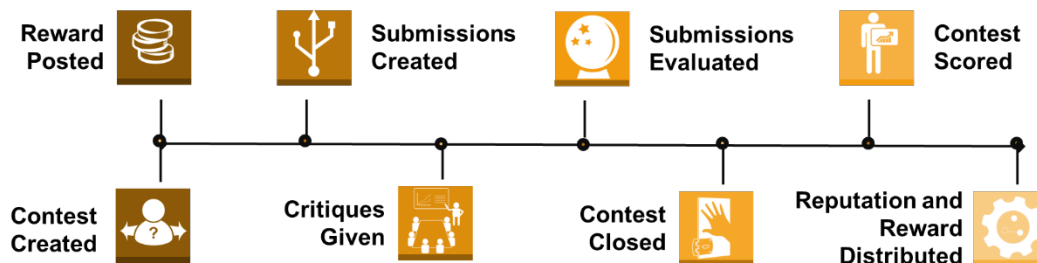


Figure 3 - Verity Contest Process

best content. Contests can be repeating, such as trying to create the best content for the day in a Reddit-like website. They can also be one time, such as a company making a contest in where experts compete to create a new product line. Verity's standard contest types all go through a standard process, during which experts compete with Creativity Tokens, Clarity Tokens, and Accuracy Tokens.

CONTEST STANDARD

All contests are implemented as smart contracts that follow a predefined standard. The contest standard determines how coins are redistributed in expert contests, initially distributed in novice matches, and how the protocol reacts to coin transfers and transformations to that contest type. This allows new crowdsourcing applications with varied needs to incorporate Verity reputation into their own unique contest types that are not covered by standard Verity contests. It also allows for various contest types to be experimented with, such as the algorithms provided by fellow crypto-reputation platforms Backfeed, Augur, Steemit, and Synereo.

MATHEMATICAL BUILDING BLOCKS

All contest types built in to Verity use essentially the same basic mathematical building blocks. The Client provides a **utility function**, which is a mathematical representation of their preferences. Then, experts try to fulfill those preferences. Their submissions are ranked on each preference using a **probability distribution**, which is a mathematical representation of their uncertainty about how the submissions should be ranked according to the client's criteria. These probability distributions might be over a **binary outcome**, spread over several **categorical options**. Or be trying to pinpoint a number in a **scalar outcome**. The probability distributions are combined using **pooling**, which is a mathematical tool to turn multiple probability estimates into one combined estimate, and oftentimes made more accurate with extremizing. A **monte-carlo method** is then used on these final probability distributions as they fit within the utility function. A monte-carlo method can be thought of as a guess-and-check method that a computer uses. Finally, a **Bayesian scoring function** is used, comparing the final values to the values each user came up with. A Bayesian scoring function is a way to compare an individual's performance to some idealized notion of performance over time. The results of this Bayesian scoring function are used to redistribute tokens. *Appendix A* defines these terms further, and provides resources to learn more.

CAMPAIGNS

Every time a Client wants a result, they create a campaign. From the Client's perspective a campaign is a single unified event in which people compete to deliver a multi-faceted solution for the Client. This is also the case for experts looking to earn Creativity tokens, whom are entering into a single contest to create the best option that matches the utility function.

However, from the perspective of experts looking to earn Clarity tokens and Accuracy tokens, each campaign will be broken up into a number of different contests centered on each input into a utility function. The results from these various contests are then combined into a single score, which gives the client a ranked list of options based on their criteria.

For most use cases, users themselves won't be creating these utility functions. Instead, the website will create a single utility function that it uses in all cases, and users will simply create and participate in contests that use that utility function.

CAMPAIGNS IN ACTION

For instance, a reddit-like site that aims to show users the best user created content might have several criteria that it wants to consider when ranking posts for users:

- An objective criterion, such as how many pageviews the post gets.
- The will of the users, expressed by upvotes and downvotes.
- An individual user's tastes, based on their past history of upvotes and downvotes.

The website would create a utility function that weights each of these three factors equally. Then, every day, the website would create a new campaign to find the best post (plus rank every other post) given that utility

function. Users would compete to submit posts that did best given those three criteria, and earn creativity tokens within that website’s community for doing well.

From the perspective of experts looking to earn Accuracy and Clarity Tokens however, there are actually three separate ways for them earn tokens

- They can predict or explain how many pageviews the post will get, and get tokens for being correct
- They can upvote or comment content that they think will be upvoted by others and get tokens for being early.
- They can choose to recommend specific content they upvote to similar users, and get tokens if those similar users end up liking it.

The reddit-like website is creating just one campaign every day, but this breaks down into multiple contests for every piece of content on the platform.

Note that the specific types of contests the users are participating in in this context represent the three types of information that a Client may want to take into account, and are explained in the following section.

CONTEST TYPES

There are three separate types of contests in Verity. With just these three types, almost any criteria one can think of can be included in a utility function. The three contest types are:

1. **Metrics Contests:** Contests in which experts compete against some objective criteria provided by an oracle.
2. **Consensus Contests:** Contests in which experts compete to please or agree with their peers
3. **Judging Contests:** Contests in which experts are evaluated along subjective criteria by a judge.





	Metrics Contest	Consensus Contest	Judging Contest	Custom Contest
				
Description	A contest that uses objective criteria to rank submissions.	A contest that uses expert votes to rank submissions.	A contest that uses subjective judgement to rank submissions.	A contest type that is created using our API
Technical Details	Each submission is ranked by a weighted scoring rule.	Each submission is ranked based on information gain from aggregate probability.	Each submission is ranked based on judges criteria.	Standard API functions to describe novice matches and contest scoring rules.

Figure 4 - Contest Types

METRICS CONTESTS

Metrics contests are contests in which experts' probabilities are ultimately judged based on real world data provided by an oracle. This oracle could be a data feed provided by a provider like Oraclize it¹³, or could even be some sort of Schelling-point oracle¹⁴ such as a Verity consensus contest. This is analogous to crowdsourcing sites like the forecasting site gjoin.com, and the data science site kaggle.com, which both use objective real-world criteria to rate their experts.

CREATIVITY TOKENS

Creativity tokens in metrics contests are awarded based on experts trying to create something which meets some objective criteria. They may be creating products which maximize sales, creating code which minimizes errors, or creating a chess strategy that maximizes chance of winning.

To redistribute the creativity tokens in a metrics contest, we use the monte-carlo simulation that plugs probability distributions into the utility function to determine how much expected utility each submission generates. The utility for each solution is divided by the total utility for all solutions and weighted by the amount of creativity tokens in the contest (in comparison to the average amount of creativity tokens in a contest). This is then used to create a weighted running mean for the user, and tokens are distributed by this weighted running mean. The creativity tokens t for expert e are calculated as:

[todo]

$$t_e = \frac{\sum_{i=1} (\frac{x_e}{x_a} \cdot t_c)}{t_a}$$

ACCURACY TOKENS

Accuracy tokens in a metrics contest are awarded based on how well the reality of a metric matches the expert's forecast. They may be predicting the sales of a product, the frequency of errors in a codebase, or the chance of a particular chess strategy leading to a win. Accuracy tokens are awarded for every accuracy along every variable that goes into a utility function. When the oracle reports the actual results, we use the Bayesian scoring rule to determine if the user's answer improves or hurts their calibration and their final Bayes score is then used to redistribute Accuracy tokens among all users.

[todo]

CLARITY TOKENS

Clarity tokens in a metrics contest are awarded based on how much an expert's comments on a particular submission helped other experts generate accurate predictions of performance. They may be explaining why

¹³ <http://www.oraclize.it/>

¹⁴ <https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/>

a particular product will sell, showing where a codebase has errors, or analyzing why different chess strategies will work.

To redistribute Clarity tokens, we first have to only show explanations to a probabilistic subset of predictors. To minimize information lost, users who already have more Clarity tokens have more of a chance of their explanations being shown. Finally, we use linear regression to determine how much impact each comment had on the Bayesian score of those predictors. These coefficients are then weighted by the amount Clarity tokens in the contest, and a weighted running mean of all the contests the user participated in is created. Tokens are redistributed based on the final spread of all these scores.

METRICS CONTESTS IN ACTION

Let's imagine how a metric contests could be used in our smart contract security community.

One place they could definitely be used would be in the auditing of smart contracts. For instance, a simple utility function could be defined which included the rating of the Dapp on the Dapp store in one year's time, and the amount of bugs found in the code in one year's time, using a clear definition of how serious something had to be to be considered a bug, and an oracle such as a Verity consensus contest.

Smart contract creators would then earn Creativity tokens by creating the most highly rated, bug free code, Smart contract critiquers could earn Clarity tokens by pointing out bugs in the code and problems with the design that would cause it to be rated poorly, and Smart Contract evaluators would predict the number of bugs and rating on the Dapp store. At the end of a year, once these values were known, reputation tokens would be distributed accordingly.

CONSENSUS CONTESTS

Consensus contests measure community agreement on a subject. They can be used as Schelling-point oracles similar to Augur's reputation metric, to rank options against each other based on their community acceptance as in Steemit, or to gauge a community's take on intangibles such as rating how beautiful a piece of artwork is. This is analogous to crowdsourcing sites such as the tech advice site StackExchange, which use consensus based mechanisms to rank their participants.

CREATIVITY TOKENS

Creativity tokens in consensus contests are awarded based on meeting a utility function just like metrics contests, with the difference being that there's no ultimate reality that decides the outcome. They can be earned by creating a forum post that the community likes, creating a mission statement that the community agrees is good, or writing code that the community determines is beautiful.

ACCURACY TOKENS

Accuracy tokens in consensus contests are awarded based on how well an expert's opinion matches the weighted opinion of a community. They may be trying to determine how much the community likes a forum post, rating how good a mission statement, or trying to determine the level of beauty of a piece of code.

To redistribute Accuracy tokens, the main scoring tool is to take the final pooled distributions, and, and compute the information gain between those distributions and the experts initial distribution, as an inverse correlation between the information gain and the final score. This is then weighted by the amount of Accuracy tokens in the contest, and computed as a running mean among all other contests that expert has participated in. The coins are redistributed based upon these final numbers.

CLARITY TOKENS

Clarity tokens in consensus contests are awarded based on how well an expert's explanation helps other experts minimize their information gain. They may be trying to explain the merits of a forum post, debate the finer points of a mission statement, or giving reasons that a given piece of code is beautiful.

Clarity tokens are redistributed much the same way as they are in metrics contest, except that instead of the Bayesian score, the inverse information gain is used as the second variable in the correlation. This score is then weighted by total Clarity tokens in the contest, and computed as a running mean among all contests that user has participated in. These scores are used as the basis for distribution of Clarity tokens.

CONSENSUS CONTESTS IN ACTION

Let's imagine how a consensus contest might be used in our smart contract security community

Firstly, one can imagine a StackExchange-like forum where programmers could discuss the creation of smart contracts. The entire site would be a daily consensus contest to ask the best question, and the questions themselves would be contests to determine the top answer. One would get Creativity tokens for asking and answering questions, Accuracy tokens for voting on them, and Clarity tokens for commenting. Instead of a single up or down vote, voting would be more like range voting, in which you could allocate however many votes to every question or answer you voted on, and the entire distribution of your votes would be interpreted as a categorical distribution, with your amount of votes indicating something akin to your probability estimate that this is the best question.

Secondly, as mentioned above, consensus contests could be used as a kind of decentralized oracle for metrics contests, using the game theoretic mechanism of Schelling Score. If you made a consensus contest for instance around how many bugs had been found in a particular contract over the past year, you'd be incentivized to count the obvious bugs that everyone else would also count, and everyone would most likely cluster around the same set of numbers at roughly the same probabilities. This allows you to have metrics contests without any worry of centralization.

JUDGING CONTESTS

Judging contests are subjective contests. If this option is chosen, a judge ultimately rates the ideas (although voters can still help to eliminate options and guide the judges). Pre-judgement scores are also weighted by similarity between the judge and the experts, in order to show the judge the proper weighting before they make their decision(s). This is analogous to crowdsourcing sites such as the design site 99designs, which use subjective criteria to choose winners.

CREATIVITY TOKENS

Creativity tokens in judging contests are awarded based on experts trying to produce content that the judge approves of. They may be trying to create a design that pleases the judge, tell a joke that the judge thinks is funny, or creating a post that a judge likes.

The redistribution of Creativity tokens is nearly identical to metrics contests with one important exception. Instead of an oracle reporting on an objective criteria, the judge reports on a subjective criteria.

ACCURACY TOKENS

Accuracy tokens in judging contests are awarded based on the experts having the same criteria as an individual judge. They may be trying to guess how much a design will please the judge, predict what a judge would find funny, or forecast which post a judge would like.

Accuracy tokens in judging contests are redistributed based on one of two criteria. In one criteria, the judge chooses a single winner, and the Bayesian scoring rule is used to score accuracy tokens. The weight of any given contest on the ultimate Bayes score of the participant is weighted based on the Honesty score of a particular client, in order to neutralize collusion. In the other criteria, the judge rates all submissions, and the inverse information gain between the judge's ratings and each expert's rating is used as the criteria.

CLARITY TOKENS

Clarity tokens in judging contests are awarded based on how well an expert's explanation helps an expert please the judge. They may be trying to explain the merits of a forum post, say why a particular design is good, or giving reasons that a given piece of code is beautiful.

Clarity tokens are redistributed much the same way as they are in metrics contest and consensus contests.

JUDGING CONTESTS IN ACTION

Let's look at how judging contests might play out in our smart contract security community.

While security seems like a mostly objective goal, with little room for subjectivity, it's nonetheless possible to imagine a scenario in which you might want to include a judging criterion in a contest. One example might be for decentralized commit access to an open source smart contract. While the ultimate goal might be full decentralization, it might be pertinent to start out with a long time contributor acting as judge, looking at things like coding standards and code clarity to make their judgements. Only once community members had learned the judge's tastes (and those who didn't eliminated from the pool) would the commit access become fully decentralized with a consensus contest.

EARNING NEW TOKENS THROUGH CONTESTS

In addition to whatever reward was posted into a contest by a client, experts also earn newly minted tokens by participating in Verity contests. These tokens are given proportionately to each contest based on how many expert tokens are in that contest, and then proportionately to each expert based on their score within that contest. These tokens are activated if the expert is below their expected tokens for the community, but

are given to the expert deactivated otherwise. Experts can choose to sell these deactivated tokens on the open market, or to keep them in hopes that they will be able to activate them in future contests.

A portion of newly minted tokens (chosen through community governance) go to experts through expert contests, and the rest goes to novices through participation in novice matches (explained below).

ECONOMIC VALUE OF TOKENS

Where Verity tokens get their economic value is an important consideration for the platform. Verity tokens having a non-zero price on the open market is important for a number of reasons:

- Allowing users to sell reputation legitimately, disincentivizes users from illegitimate methods of profiting on reputation that can destroy the predictive power of it, such as selling their account.
- Requiring users to buy into similar communities isolates each community from attacks on adjacent communities, creating a cost for every subsequent community the user would like to attack.
- The price of reputation creates an incentive for users to participate in communities that may be socially but not economically lucrative such as non-profits, by giving them the chance to sell their tokens to people whom are looking to prove those same skills in for-profit ventures.

This means that it's important to do some reasoning about the economic properties of reputation token, and where they may derive their value.

PRICING ACTIVATED TOKENS

An activated token is any token which a user purchases while still below their expected tokens for the community they're purchasing from. The central reason that a user would want to purchase activated tokens instead of earning them on their own is to be able to get more rewards faster from participating in contests.

Therefore the price that an individual expert should be willing to pay for tokens is determined by the net present value of the rewards that an expert would get when buying the activated tokens directly minus the net present value of the rewards that an expert would get over that same period it would take them to earn those tokens through competing in contests. Thus the relevant factors for reputation price are:

- How long it takes to earn new reputation tokens
- How economically valued the reputation token is within contests.

This seems to match our intuitions about how reputation should be valued – Reputation that is harder to earn and more economically valuable tends to be seen as more valuable in society.

PRICING DEACTIVATED TOKENS

When looking to price deactivated tokens, similar considerations are taken into account. However, there are two additional caveats. Firstly, the amount of time that can be shaved off by buying deactivated tokens is significantly reduced because getting the benefit from them requires participating in contests. Secondly, there's a third factor besides the hardness and economic values of contests – there's the users probability estimate of how many reputation tokens they'll earn and lose in that period.

For the purposes of clarity, make three simplifying assumptions. Firstly, it will take users twice as much time to earn tokens the normal way as it will to earn tokens by buying them deactivated. Secondly, this means the user will be able to earn double the rewards over the same time period that it would take them to earn those tokens the normal way. Thirdly, their ability to earn those tokens is a simple binary situation where the user either gains all the deactivated tokens with some probability, or loses all the deactivated tokens.

In this case, the user should be willing to pay double the net present value of earning the tokens the normal way, multiplied by their own confidence interval that they will indeed earn all those tokens. Once again, this matches with our intuition of how reputation should be bought and sold in the real world, such as in the case of an individual buying a brand in the assumption that they'll be able to maintain the same value as the previous owner.

PRICING VERITY CLEAR TOKENS

The price which an expert is willing to pay for Verity Clear tokens should track the price of the highest price activated token at any given time, as they can be turned into these tokens at no cost. However, this assumes that the value of buying token in an existing community will always be more than the value of creating a new community.

Any new website that wishes to use our technology in an entirely new community (and therefore get most of the early stake within that community) will want to purchase Verity Clear in order to seed that community with some experts. Thus the value that a community creator would be willing to pay is determined by the net present value of the expected future profits from that community (which they can get either by being an expert in that community, or a client.)

Thus whether or not Verity Clear tokens are being bought for community creation or reputation will fluctuate. As new use cases for Verity are created, it will be more attractive to create new communities to take advantage of those use cases. As the websites behind those communities begin to grow, it will be more attractive to transform the tokens into reputation tokens. Then, as the communities mature, this will in turn enable new use cases, which will again create the incentive to create new communities.

ONBOARDING USERS IN VERITY

Verity has two distinct ways to earn it's tokens.

1. **Novice Matches**, involving contestants who are working to earn newly minted Verity colored tokens.
2. **Expert Contests**, involving expert contestants gaining and losing tokens from each other, in addition to earning newly minted and possibly deactivated tokens.

By separating the two ways to earn tokens, we have a smooth path for which users can work towards earning money on the platform while learning community norms and improving their skills.

NOVICE MATCHES

Novice Matches allow new users to grow their affinity score and earn Verity tokens as a new member of a community. They're scored very differently from expert contests, and must use a separate Sybil defense mechanism because they can't use token ownership as a proxy. Every contest contract can specify a corresponding novice match protocol, and if so, will have a mirrored version of all expert contests using that protocol. The general format of novice matches is as follows

1. Only accounts without Verity Tokens in that community can participate in the matches
2. All accounts must complete a nominal task such as a captcha to participate in matches for that month.
3. Only accounts with an affinity score of .5 can earn token in those matches.
4. Novice Matches are played out over the course of a month. At the end of the month, all qualified participants are given a percentage of the newly minted Verity tokens, based on their performance in the novice matches that month.

For Verity's three contest types, the Novice Matches play out very similarly to how the expert contests described above play out. The difference being that instead of coins being used to determine the level of someone's influence for Accuracy tokens, an influence limiter function using a reputation score is used before the Bayesian Scoring function as described by Resnik and Sami in their paper *The Influence Limiter: Provably Manipulation-Resistant Recommender Systems*¹⁵.

The central insight for using an influence limiter is the same as that used to rate consensus contests - a user should raise their reputation if weighing them more would have created a smaller information gain in previous predictions.

[todo – algorithm for Resnik's influence limiter]

Because there's no notion of pooling influence to create Sybil attacks for creativity tokens or clarity tokens, those are scored much the same as described for expert contests, with the amount of coins in the expert contest being used for the weight, and everyone starting with some nominal score like .001.

EXPERT CONTESTS

Once an expert has accumulated some tokens, they can enter expert contests. Expert contests are the only way in which experts can be paid by Clients on the system, and are used as the basis for which winning entries are presented to the Client.

All expert contests in Verity follow the same basic format. First, experts compete to meet the goals of the campaign as defined by the Client. Based on how well the expert does, they may earn reputation from other

¹⁵ P. Resnick and R. Sami, "The influence limiter: Provably manipulation resistant recommender systems," in *ACM RecSys '07*, pp. 25–32, ACM, 2007

experts, or lose reputation to those experts. Then, based on how much reputation they have when the contest closes, they earn a portion of the fee that the Client paid for the contest deliverables.

Because influence in expert contests is limited by tokens, they already have an effective Sybil defense, which holds below 50% of contest participants being Sybil accounts.

CONTEST RULESETS

Any community can choose a skill tag that it chooses to measure using its reputation tokens, provided it has some Verity clear tokens that it can seed its members with. The community simply chooses what **contest ruleset** it would like to allow to for its skill tag, and a variety of different behaviors emerge. To start with, Crystal will provide three sets of rules around contests, which allow three very different options for what skills mean. These three different rulesets allow a very flexible definition of skills that can encompass most use cases.

GUILD SKILLS

A **guild skill** in Verity allows any contest that has consistent values, but a customizable utility function. In practice, what this means is that the community defines what type of expertise it would like to represent, and then it allows any client to have its members fulfill that client's goals. The guild's challenge in this case is to define their values in such a way that they believe strikes a good balance in flexibility of clients vs. specialization of skills.

A real life example of the guild skill model would be 99designs.com. 99designs handpicks what it calls "Platinum Designers" – designers whom meet the criteria that 99designs values in designers (99designs, 2016). At the same time, 99designs allows for clients to choose their own criteria, using what it calls "style attributes" to describe what features the client is looking for.

Instead of handpicking designers one by one, with Verity one could imagine a model where 99designs handpicked a few people it considered "good designers," and use this as the start-set for the "good design" value. New platinum designers could then grow organically as members within the community rated each-other with that good design value.

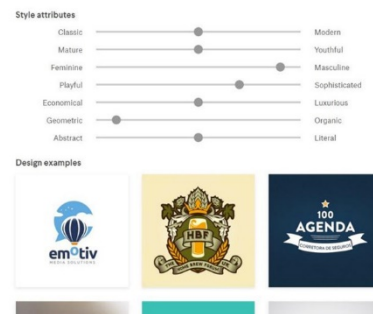


Figure 5 -Style Attributes in 99Designs

SHARED SKILLS

A **shared skill** is a flexible skill type in which there's some broad agreement around what goals a skill is trying to accomplish, but there many different values that different communities have that further refine that skill for their particular community. In practice, what this means is that the community defines a standard utility function for every contest, but allows each contest to specify its measurement for affinity, by which the skill tokens get weighted for that contest.

While there are currently no real life examples of shared skills, this is simply because no system like Verity currently exists. An example of where a shared skill might be useful is for a skill tag called “internet commenting.” A simple utility function could be defined, which would be a consensus contest where the goal was to up-vote comments that other members also up-voted. This same skill could be shared by Huffington post, The New York Times, Reddit, and YouTube.

In order to capture the different types of comments that might be considered valuable on different platforms, an individual’s tokens can then be weighted by affinity to each community’s own values. For instance, humor might be valued very highly in Reddit, but lower in The New York Times. Alternatively, YouTube and reddit might both value humor, but have different ideas of what constitutes good humor. This mechanism captures these differences quite elegantly.

ORGANIZATIONAL SKILLS

Organizational skills are skills in which both the goal and the values are fixed. In practice this means that an organization has its own utility function, and people can prove their value to that organization by having values the organization values, and helping it maximize its utility. The tag will be something like ‘Adding Value to Verity’, and the amount of tokens an individual holds would represent how valuable they are to the Verity organization.

An example of how this works in the real world is pay scales. An organization tries to find people that fit in with its culture, then pays them based on how much value they add to the organization.

CONTRIBUTION-BASED REPUTATION

Contribution-based reputation is the third core type of reputation. While organizational skills-based reputation measures an expert’s skill at adding value to an organization, contribution-based reputation actually measures how much value that person has added – that is, it takes into account the amount of work actually done. By using Verity contests to create clear measures for each action, a transparent, accurate method of ranking contributions emerges.

UTILITY FUNCTION STANDARD

In order to issue contribution-based reputation, a community must first choose

A utility function is a function written in solidity that outputs a number. It indicates the community’s preferences and goals – the higher the number, the more the community wants that outcome. Contribution tokens are handed out based on how well individuals meet these utility functions.

A community’s utility function is chosen according to the governance mechanisms in that community, and can evolve over time as the communities goals and preferences change, or the community realizes that their initial utility function didn’t fully specify those goals and preferences.

All utility functions follow Verity’s utility function standard – a standard framework that can be used by any governance mechanism that requires mathematical precision around goals (such as futarchy).

The utility function standard gives a serialized format that explains all input variables, a simple function to plug all relevant variables in that returns the utility, as well as a standard functions to deal with specialized utility (explained below).

SPECIALIZED UTILITY

One common thing that communities want to do is measure specialized tasks, actions or roles. While a general utility function is needed at the community level, it may make sense to create customized utility functions that relate a community's goals to specific metrics or actions in an individual role. For instance, it may make sense to measure a phone support rep by customer satisfaction and calls/hour, and then have a standard way to relate these metrics to the broader community goals.

To handle this, Verity allows one to create specialized utility functions that relate to a specific task, and relate directly back to broader community goals. This can be used to create individual metrics for specific roles, to give standard rewards for things like referring new members to the community, or even give a standard one-time reward for doing something like reading the employee manual.

CONTRIBUTION TOKENS

Contribution tokens are used to measure how well an individual helps a community meet its utility function. Contribution tokens work by assigning a value to actions that community members take.

CALCULATING CONTRIBUTION TOKENS

Contribution tokens are calculated by using Verity contests to measure the utility of a contribution. In the absence of contests, specialized utility can be used to give contribution tokens for certain actions as well. When a contest is complete, contribution tokens are minted and issued in accordance with how much utility a specific contribution gives to the community. Optionally some amount of contribution tokens can also be issued for accuracy, and critiques (using specialized utility) which are then distributed according to the rules of the contest.

GOVERNANCE

Verity's base governance layer at its core is a simple standard that can be extended with smart contracts to encompass arbitrary governance protocols. This standard defines governance for each individual Verity Community, as well as governance at the protocol level, called VerityDAO, which defines global rules about interactions between and within every community

On top of this standard we build three primary forms of governance. The first is a traditional hierarchical structure. The second is decaying-contribution voting, which is a democratic voting mechanism that gives more weight to those who are offering more value to the community. The third is Reputarchy, which uses voting to set the goals of a community, then allows experts who have proven they're good at fulfilling those goals to make decisions.

For individual communities, governance is decided at the point of creation. The plan for the VerityDAO is to start with a hierarchical structure, in which the decisions are informed by Reputarchy, and build in incentives (similar in spirit to Ethereum's difficulty bomb¹⁶) to switch to Reputarchy as the ecosystem of expert communities matures.

GOVERNANCE STANDARD

Verity's governance standard is a simple flexible standard that includes hierarchical roles, permissions, actions, and action states. The systems also allows roles to be held by smart contracts, which themselves can follow the governance standard. By layering these simple building blocks together, complex governance mechanisms can be built, and mechanisms like Futarchy, Reputarchy, or Quadratic Voting can be easily swapped in and out as desired. Nexusdev's dappsys framework¹⁷ already includes much of this functionality, and we're currently investigating the possibility of adapting it for our purposes.

ACTIONS

An **action** as defined by Verity is a specific function that lives within a smart contract and can be executed on the Ethereum network. The action could be a simple action that merely triggers an event that shows that the action happened, or can be a complicated action that involves invoking outside contracts or moving ether. The action may take arguments, which are inputs to the action that change its meaning, or they may simply be actions that have one meaning, such as "resign."

An example of an action in the US government might be "approve citizenship" which takes as an argument the identity of a potential citizen.

PROPOSAL STATES

A proposal state represents a proposal's status within the governance process. Every action has two states by default: 'rejected' and 'approved'. A rejected proposal is a proposal which is dead, and can longer change states. An 'approved' proposal is a proposal which will be executed on the blockchain. In Verity's governance standard, any number of arbitrary states may be created representing arbitrarily complex governance rules.

For instance, in the US government, when the action proposed is "create bill" (and the bill itself is the argument), the proposal would go through the "senate vote" state, the "house vote" state, and the "presidential veto period" state, before finally being accepted or rejected.

PROPOSALS

A proposal in the governance standard is a suggested action (with suggested arguments) that has not yet been executed. A proposal can go through any number of states before being executed or rejected.

¹⁶ <http://ethereum.stackexchange.com/questions/3779/when-will-the-difficulty-bomb-make-mining-impossible>

¹⁷ <https://github.com/nexusdev/dappsys>

ROLES

A **role** in the governance standard is a specific function that can be assigned to any agent in the system. For instance, roles in the US government might include Citizen, President, Senator, and Congressman. Other non-obvious roles might include The Constitution, The Senate, and the House; these roles would be filled by smart contracts, which themselves could implement their own governance rules.

Verity's standard also supports hierarchical roles. A child role will inherit all permissions from its parent role unless otherwise specified, as well as any additional permissions assigned to it. For instance a CEO can inherit at the very least all the permissions that a manager has, and a manager can inherit all the permissions of an employee.

PERMISSIONS

Permissions in Verity are given to specific roles and give that role the ability to participate in certain ways within the governance system. There are only two types of permissions within the governance system; roles can be given the permission to execute an action, and they can also be given the permission to change a proposal from one specific state to another specific state.

In the US government, a "senator" role might have permission to execute a "vote in the senate" action, and a "the senate" role might have permission to change a "create bill" proposal from the "senate vote" state to the "house vote state" or the "rejected" state.

GOVERNANCE SYSTEMS

While Verity can represent any governance system (that can be represented in a Turing complete language), it will come with several options out of the box, which make unique use of Verity's own reputation metrics. These reputation-based governance methods represent new paradigms that can work equally well in traditional settings, as well as decentralized applications.

VALUE-BASED HIERARCHY

In a value-based hierarchy, every role in an organization is matched with a value tag, such as "good customer support person," "good customer support manager" or "good engineer." The roles exist in a hierarchy, such that executives exist above customer support managers, and customer support managers exist above engineers.

The central idea behind value-based hierarchy is that each stage in the hierarchy gets to choose the start-set for the role below it. Thus executives choose exemplary examples of managers, managers choose exemplary examples of employees, and so on. When someone's value score rises above .5 for a given role, they are automatically given all the permissions and perks associated with that role.

This allows the entire organization to participate in the promotion process and be on the lookout for ideal employees, while still giving higher-ups the ability to choose what type of qualities they're looking for in each role, and prizing the opinions of individuals already in the role.

CONTRIBUTION-BASED VOTING

In contribution-based voting, individuals vote on actions, and their votes are weighted by the amount of contribution tokens that individual holds within that community. This can be seen as a meritocratic system in which those whom offer more value into the community get more say within it. This voting method can optionally have contribution tokens' weight decay logarithmically over time, such that people whom have given more recent contributions are given more weight than those whom gave contributions long ago in the organizations history.

SKILL-BASED REPUTARCHY

Reputarchy uses Verity's reputation and community mechanisms to give decisions to those whom have proven they're best at making them. If futarchy¹⁸ is described "vote on values, bet on beliefs," then Reputarchy could be described as "vote on values, crowdsource beliefs." The community (potentially using contribution-based voting) votes on the goals and value of the communities, and then decisions are made by creating Verity contests that maximize those goals and values.

REPUTARCHY STEP BY STEP

Reputarchy follows a three step process, in which the community chooses a utility function, chooses actions to undertake based on that utility function, and then implements those actions.

CHOOSING A UTILITY FUNCTION

A key part of reputarchy is choosing a utility function. There are three ways that the values and goals in reputarchy could be implemented in a utility function using Verity. Firstly, they could be an entirely subjective set of values and goals described in plain language and voted on by constituents, given to experts to both interpret and implement using consensus contests. Secondly they could be an objective set of metrics voted on by constituents, given to experts to optimize then measured using objective contests. Thirdly, they could be a subjective set of metrics combined from constituents using for instance a Likert Scale, then measured using judging contests.

CHOOSING ACTIONS TO TAKE

Once a utility function is chosen, the next step is to choose a number of skill-tags, and communities known to be good measures for those skill tags in a variety of different disciplines. At regular intervals (weekly, monthly), a contest will be held in each of these communities, asking "What's the best thing we should do around your expertise to maximize our utility?" The utility of every idea in every community will be compared, to find the ideas that have the best chance among all actions in all contests of fulfilling the organization's utility will be selected

¹⁸ <http://mason.gmu.edu/~rhanson/futarchy.pdf>

TAKING ACTION

Finally, the actions need to be implemented. Requests for proposal will be created for each action determined in the previous steps, and the proposals themselves will be submitted into a Verity contest. The proposals can be submitted by traditional contractors, or Verity communities themselves, who will crowdsource the work using contests. All possible proposals will be ranked against their utility in the utility function, and the money to implement the action will go to the winner.

REPUTARCHY IN ACTION

Imagine an “end global warming” community that made a utility function with two variables: how much carbon emissions are lowered within five years, and how ethical each action is considered by ethical members of the community. It would then choose guilds and shared skills which it thought it might need, such as choosing the “Washington lobbyists” guild skill for the “lobbying” skill tag and the “high-tech inventors” shared skill for the “technology development” skill tag. The community could then create a contest for the best lobbying strategy that met its utility function of lowering carbon emissions, and have the inventors work together to come up with potential promising approaches to reducing carbon emissions. It would choose those actions which had the highest overall reduction in carbon emissions, and hire contractors to execute on those plans.

USE CASES OF REPUTARCHY

While reputarchy will likely first be implemented in Verity communities, the real power of it comes when other DAOs and traditional organizations begin to implement it. This will mark a shift at which organizations will have to be upfront about exactly what they’re optimizing for, and individuals get paid only for their performance. This will severely limit the ability to play politics within an organization.

VERITYDAO

The VerityDAO is the organization which is in charge of running the central protocol and overseeing the growth and maintenance of the entire Verity ecosystem. VerityDAO plans to start with a simple multisig governance mechanism, informed by Reputarchy style suggestions. then later switch to fully Reputarchy-controlled governance mechanism as the ecosystem of expert communities matures.

MAKING A PROFIT

VerityDAO makes money by taking a small cut of the reward from every paid contest in every community. This reward then goes into the Verity smart contract, where it can be used to help fund new communities, bolster existing communities, create better infrastructure, or be passed on to vote holders in the form of weekly payments.

REFERRAL INCENTIVES

Every user on the Verity platform can receive referral incentives for referring other active users who add value to the Verity platform, whom also get an incentive for signing up under their friend. An active user is anyone whom earns activated skill-tokens in a pre-existing community. Both the referrer and the referred receive a reward of Verity Clear tokens. The VerityDAO chooses how much this reward is, and can also set it to zero.

VERITYDAO GOVERNANCE ACTIONS

As the parent organization, VerityDAO is in charge of taking actions that cause the ecosystem to flourish. What follows is a list of some of the most important actions that VerityDAO can take

SETTING VERITY TOKEN ISSUANCE RATE

Issuance of new tokens in Verity is set at a steady monthly limit, which can be tweaked by the stakeholders. This is analogous to a central bank which is incentivized to maximize the value of the tokens. Newly issued tokens are split percentage wise among all the communities, and are created to fuel novice matches and allow the onboarding of new users into Verity communities.

SETTING VERITY TOKEN BUY AND BURN RATE

Just as it may be prudent to issue more tokens in order to onboard more users onto the platform, it may also make sense to shrink the supply in order to raise the value of Verity tokens. For this purpose, the Verity has an ongoing offer to buy and burn tokens at a price set by the DAO. While this price will typically be set to 0, it may be raised above market value if the supply is growing too fast, in order to incentivize burning of tokens and shrink the supply.

CHOOSING CORE VALUES

The VerityDAO is in charge of choosing its own core values. These core values will in turn determine the value-weight of all votes in the parent organization, based on those voters' affinity scores with the organization's core values.

ADDING AND REMOVING MEMBERS FROM THE START SET

For each of the core values, the VerityDAO can choose members whom they view as paragons of those values, and add them to the start set. If it comes to light that a person has a different character than previously assumed, they can be removed from the start set.

SETTING CONTEST FEE

Every contest on the Verity platform has a small percentage fee taken from it, which goes to the parent organization. The VerityDAO sets this fee at such a level to be competitive, while still allowing maintenance and growth of the platform.

SETTING STAKE-WEIGHTED PAYMENTS

Every week, a small amount of earnings are taken from the VerityDAO and issued to stakeholders, weighted by their stake. The VerityDAO votes on this amount.

UPDATING THE PROTOCOL AND GOVERNANCE RULES

The VerityDAO is in charge of creating updates to the protocol, as well as to its own governance rules.

SENDING MONEY

The VerityDAO can send money to any account on the Ethereum platform. This may be in payment to a contractor, as a stimulus to an individual community, or for any other reason.

COMMUNITY GOVERNANCE

Communities are places where experts can gather, converse, and work together to provide value to each other or to clients. Every community has different goals and purposes, defined by their members and core values.

CREATING A COMMUNITY

Communities are created in stages.

First, the idea for a community is proposed. Potential participants in the community will begin to interact with each other, tagging each other with value tags as they discuss the community.

Then, some group of founding members will then initialize the community by choosing a governance mechanism, and choosing the initial rules for every aspect of the community. Core values can be chosen by looking at commonly tagged values between founding members.

From there, a community defined initialization period commences during which any person can commit Verity Clear reputation to the system, turning it into initial reputation for the community at 100% activation rate.

After the initialization period, the community starts, and people begin to start creating and participating in contests. At this point, the community rules and values can be changed according to the chosen community governance protocol. Other Verity tokens can enter the community by transforming according to the normal rules.

COMMUNITY GOVERNANCE ACTIONS

CHOOSING CORE VALUES

The community is in charge of choosing its own core values. These core values will in turn determine the value-weight of all votes in the parent organization, based on those voters' affinity scores with the organization's core values.

ADDING AND REMOVING MEMBERS FROM THE START SET

For each of the core values, the community can choose members whom they view as paragons of those values, and add them to the start set. If it comes to light that a person has a different character than previously assumed, they can be removed from the start set.

SETTING CONTEST FEE

Every contest in the community has a small percentage fee taken from it, which goes to the community. The community sets this fee at such a level to be competitive, while still allowing maintenance and growth of the community.

CHOOSING PRIVILEGED CONTEST TYPES

Every community chooses some subset of contest types that it considers privileged. These contest types are allowed to distribute new tokens through novice matches, and also distribute stake to the VerityDAO through participation.

UPDATING CONTESTS AND GOVERNANCE RULES

The community can choose to update its privileged contests to new versions, as well as changing its governance contract.

SENDING MONEY

The community can send money to any account on the Ethereum platform. This may be in payment to a contractor, or for any other reason.

USE CASES

DECENTRALIZED APPLICATIONS

Decentralized applications (Dapps) have a particular problem that other applications don't have. They often need to make decisions that require human judgement, but giving the decision to any one person or small group of people can end up re-centralizing that app by concentrating power. With Verity, all of these human judgements can be made in decentralized way. For instance, one can **moderate content** without a centralized moderator, **settle user disputes** without a single company to handle support, **identify trusted nodes** for distributed work, and even create **truth oracles** that provide trusted data without relying on a single source.

RATING AND GRADING

Ratings and grades are notorious for creating adverse incentives that cause bad behavior. The centralized review site Yelp has been accused of extorting business to bury negative reviews¹⁹, and teachers have been known to give their students the correct answers on standardized tests²⁰. With Verity, you can **democratize education** through decentralized assessments, create **cheaper insurance** through crowdsourced risk assessments, and **disintermediate review websites** with reviews and ratings that can't be gamed.

COLLABORATIVE CROWDSOURCING

Anything that can be created by an individual on a computer, can be created by Verity crowdsourcing. With Verity, the things built are optimized to meet your exact goals or preferences, and are typically better results than you would receive by contracting a single expert. You can **have software reviewed by thousands of reviewers**. You can create a **compelling creative book**, and give royalties based on how much each person contributes. You can even **design cars, buildings or organizations**, all at a cheaper price point, and likely greater quality, than expert work.

PORTABLE REPUTATION

With Verity, you can earn reputation once, and take it everywhere. This means that **low quality internet comments will be buried**, as the best commenters earn their reputation and take it from site to site. Instead of needing to prove yourself on new **marketplaces**, you can take it with you from e.g. Amazon to eBay. Finally, the **sharing economy** will be so much safer. Your Uber, Airbnb, and Snappgoods reputation will all be linked, allowing every user to get a full measure of the service they will receive.

DECISION MAKING

With Verity, making decisions will be decision-theoretically optimized according to your goals. Want to get **individual advice** on buying shoes that look great and last forever? With Verity, the crowd will find the perfect pair for you that balances those two goals. **Business strategy** will forever be changed as companies find out that their CEOs' decisions can be matched or beaten by communities, with lower price points. Eventually, **countries** may even jump on the reputarchy bandwagon.

FORECASTING

Knowing future possibilities is crucial to wellbeing of millions of people. Wouldn't it be great to know the likelihood of a major war springing up in the next year or the chance of an earthquake in California in the next six months? With Verity, **large events, natural disasters, and existential risks**, can all be given precise

¹⁹ <http://www.huffingtonpost.com/news/yelp-extortion/>

²⁰ <http://abcnews.go.com/US/atlanta-cheating-178-teachers-administrators-changed-answers-increase/story?id=14013113>

probabilities. These probabilities have been shown to outcompete even the top forecasting experts²¹, and are of similar quality to prediction markets²².

MACHINE LEARNING AUGMENTATION

Training machine learning systems to interact with human systems is a common goal, but the existing approaches suffer from limited training data and poor specification of human values. Verity can solve these problems and create a synergistic effect between machines and humans. Using the Verity protocol, for the first time there's an **API for human ingenuity**, which communities can be plugged into in order to solve problems machine learning algorithms can't. The machine learning algorithms can then use these **contests as training data**, significantly reducing the effort needed to compile such a data set. Finally, these machine learning algorithms can be used to further assist humans, completing the cycle. In general it is assumed that participants in Verity communities can be human, AI or humans augmented by AI.

CONCLUSION

For the first time, Verity makes possible credible and immutable reputation metrics that represent the attributes we care about in the real world. Our value rank algorithm allows any humans' values to be quantified using the notion of transitive trust. Our Verity tokens allow any humans' skills to be quantified using basic decision-theoretic criteria.

By combining this reputation with a novel method for flexible crowdsourced contests, we can harness the wisdom of the crowd to out compete experts. This method has far reaching implications for governance, decision making, and society at large.

BIBLIOGRAPHY

- 99designs. (2016, October 28). *How to become a Platinum designer on 99designs*. Retrieved from 99designs Blog: <https://99designs.com/blog/inside-99designs/platinum-designer-99designs/>
- Aggregative Contingent Estimation*. (2016, September 30). Retrieved from IARPA: <https://www.iarpa.gov/index.php/research-programs/ace>
- Culbertson, S. S., Henning, J. B., & Payne, S. C. (2013). Performance Appraisal Satisfaction: The Role of Feedback and Goal Orientation. *Journal of Personal Psychology*, 189-195.

²¹ <http://www.npr.org/sections/parallels/2014/04/02/297839429/-so-you-think-youre-smarter-than-a-cia-agent>

²² http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2660628

- Fan, X., Liu, L., & Li, M. (2012). EigenTrust++: Attack resilient trust management. *2012 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)* (pp. 416-425). Curran Associates.
- Galton, F. (1907). Vox Populi. *Nature*, 450-451.
- Google Scholar Search. (2016, October 30). Retrieved from Google Scholar:
https://scholar.google.com/scholar?ion=1&espv=2&bav=on.2,or.r_cp.&biw=1366&bih=667&dpr=1&um=1&ie=UTF-8&lr&cites=4085997296011241419
- Gun Sirer, E. (2016, October 17). *Micropayments an Cognitive Costs*. Retrieved from Hacking, Distributed:
<http://hackingdistributed.com/2014/12/31/costs-of-micropayments/>
- Resnick, P., Zeckhauser, R. J., Swanson, J., & Lockwood, K. (2006). The Value Of Reputation On Ebay: A Controlled Experiment. *Experimental Economics*, 79.
- Spiegel, A. (2016, September 30). *So You Think You're Smarter Than A CIA Agent*. Retrieved from NPR:
<http://www.npr.org/sections/parallels/2014/04/02/297839429/-so-you-think-youre-smarter-than-a-cia-agent>
- Szabo, N. (1999). Micropayments and Mental Transaction Costs. *2nd Berlin Internet Economics Workshop*.
- Sztorc, P. (2016, September 12). *Oracles are the Real Smart Contracts*. Retrieved from Truthcoin Bog:
<http://www.truthcoin.info/blog/contracts-oracles-sidechains/>
- Traupman, J. (n.d.). *EECS Technical Reports*.
- Vogelsteller, F. (2016, September 30). *ERC: Token standard*. Retrieved from Github:
<https://github.com/ethereum/EIPs/issues/20>

APPENDIX A - MATHEMATICAL BUILDING BLOCKS

UTILITY FUNCTIONS

A utility function is a mathematical representation of preferences. In decision theory, they're central to the concept of making sound decisions, and for that reason they're the basis of Creativity tokens in Verity. One's creativity will be scored based on how well one can generate options that maximize a given utility function. In addition, for each input into a utility function, a contest will be created to earn Accuracy and/or Creativity tokens.

Because there are near limitless representations of utility functions, Verity must allow for the possibility of arbitrary complexity. Given this constraint, each utility function will be its own smart contract on the

Ethereum platform. These smart contracts will have standard functions, which input any number of variables through a contest format, and output a number. The higher the number, the more utility received.

Utility functions are hard to grasp for the average users, and we're actively looking into easy ways to intuitively create utility functions, including LENS modeling, willingness to pay models, visual equation builders, and premade utility function libraries. It's also worth noting that for many applications, utility functions won't have to be created by the user, as the utility function will be premade by the front-end dapp, as in our examples with smart-contract security.

SCORING RULES AND DIVERGENCE MEASURES

A scoring rule is a rule which incentivizes participants to be accurate to create good forecasts by giving higher score to more accurate forecasts. A strictly proper scoring rule is any scoring rule in which incentivizes participants to be honest about their true probabilities, by giving the highest score to the most accurate predictions. In 2008, two families of strictly proper scoring rules were found to be able to incorporate decision-theoretic notions of risk-tolerance, information-theoretic notions of information gain, and approximate all of the popular scoring rules²³. These constructions were extended to incorporate the property known as sensitivity-to-distance which allowed forecasters whom were closer to the correct answer to receive higher scores, as well baseline distributions, which allowed for the notion of prior information. The formulas for computing the pseudospherical and power rules with baseline distribution q are as follows:

$$\text{Power Rule with Baseline: } S_{\beta}^P(r, e_j \| q) = \frac{(r_j/q_j)^{\beta-1} - 1}{\beta-1} - \frac{\sum_{i=1}^n (r_i(r_i/q_i)^{\beta-1}) - 1}{\beta}$$

$$\text{Pseudospherical Rule with Baseline: } S_{\beta}^S(r, e_j \| q) = \frac{1}{\beta-1} \left(\left(\frac{r_j/q_j}{(\sum_{i=1}^n (r_i(r_i/q_i)^{\beta-1}))^{1/\beta}} \right)^{\beta-1} \right) - 1$$

Where β is any real number, r is a reported distribution, e is an event, with j being the actual outcome and i being the reported outcome, n is the total number of outcomes, and q is the baseline distribution.

Each can be extended to include sensitivity to distance with the following equations.

$$\tilde{S}_{\beta}^P(r, e_j \| q) = \sum_{i=1}^{j-1} S_{\beta}^P(R_i, e_2 \| Q_i) + \sum_{i=j}^{n-1} S_{\beta}^P(R_i, e_1 \| Q_i)$$

Power rule with baseline and sensitivity to distance.

[todo – pseudospherical rule + information measures]

²³ https://faculty.fuqua.duke.edu/~rnau/scoring_rules_and_generalized_entropy.pdf

The incorporation of a baseline distribution also allows the possibility for emulation of market scoring rules²⁴, a creation of Robin Hanson which use previous information gained from forecasters as the baseline.

By incorporating both the discrete and continuous cases of these two constructions, we can allow every contest to use the scoring rule that best suits their needs, whether through a website choosing this option for their users, or by creating a wizard through which a client can choose the scoring rule that best fits their needs.

POOLING ALGORITHM

The pooling algorithm is the algorithm which combines probability distributions into a single distribution. A pooling algorithm can be as simple as a weighted average, known as linear pooling, with Accuracy tokens used for the weight. There are several other algorithms which are shown to be more accurate by engaging in a process known as extremizing. An active area of our research is around the tradeoffs to be made in terms of simplicity, gas costs, and accuracy in different pooling algorithms.

MONTE CARLO SIMULATION

A Monte-Carlo simulation for our purposes is a way to brute-force input the results from an aggregated probability distribution into a utility function. It can then be used to show how different submissions rank in regards to that utility function. Monte-carlo methods are very expensive, and it may be that we cannot find a suitable method that also costs sufficient gas. If this is a case, a “shadow chain” will be used, a concept first described in a blog post by Vitalik Buterin²⁵.

APPENDIX B – MATH FORMULAS AND ALGORITHMS

B.1 - EIGENTRUST ALGORITHM

1. A **local trust value** is computed for every node the originating node interacts with. This is done by taking the number positive interactions with that node minus the number of negative interactions.

$$s(i,j) = sat(i,j) - unsat(i,j)$$

2. A distribution is defined in which each member of the start-set is given equal trust.

$$p(i) = \begin{cases} 1/|P| & \text{if } i \in P \\ 0 & \text{otherwise} \end{cases}$$

B.1 - Definitions

$s(i,j)$: Satisfaction measure of transactions peer i had with peer j
 $sat(i,j)$: Number of satisfactory transactions peer i had with peer j
 $unsat(i,j)$: Number of unsatisfactory transactions peer i had with peer j
 $c(i,j)$: Normalized local trust value that peer i assigns to peer j
 P : The start-set of pretrusted peers
 $p(i)$: The start trust put into an individual peer i from the set P of trusted peers
 $t(i,k)$: The propagated trust value that peer i assigns to peer k
 \rightarrow_p : The distribution of trust over the set P of trusted peers.
 C : The matrix $[c(i,j)]$
 \rightarrow_t : The left principal eigenvector of C

²⁴ <http://mason.gmu.edu/~rhanson/mktscore.pdf>

²⁵ <https://blog.ethereum.org/2014/09/17/scalability-part-1-building-top/>

3. The local trust score is normalized over all local trust scores of nodes that this node has rated, such that all the local trust scores sum to 1, with any trust scores below 0 being dropped entirely. If the normalized trust score is 0, the node will spread it's trust among the start set.

$$c(i,j) = \begin{cases} \frac{\max(s(i,j), 0)}{\sum_j \max(s(i,j), 0)} & \text{if } \sum_j \max(s(i,j), 0) \neq 0 \\ p(j) & \text{otherwise} \end{cases}$$

4. These normalized local trust scores are then aggregated, by assuming that trust is transitive. For a peer k that peer i has never met, peer i will add the normalized trust scores of all peers j whom have met peer k , and weight those scores by peer i 's own trust in peer j .

$$t(i,k) = \sum_j c(i,j) \cdot c(j,k)$$

5. This process is continued outward, such that peer k becomes the new peer j . Eventually, the node has a complete view of the network. This can also be viewed as a probabilistic process, in which following nodes with higher t 's will result in landing on a more trustworthy peer. This markov process can be modeled using linear algebra.

$$\vec{t}^{(0)} \Rightarrow \vec{p}$$

Repeat

$$|\vec{t}^{(k+1)} = C^T \vec{t}^{(k)};$$

$$|\vec{t}^{(k+1)} = (1 - a) \vec{t}^{(k+1)} + a \vec{p};$$

$$|\delta = \|\vec{t}^{(k+1)} - \vec{t}^{(k)}\|;$$

Until $\delta < \epsilon$;

B.2 – EIGENTRUST++ ALGORITHM

1. Average all local trust ratings together, between all nodes w that nodes i and k have both rated

$$t(i,w) = \frac{\sum_{i=1}^n t_i(i,w)}{m} \text{ and } t(k,w) = \frac{\sum_{i=1}^n t_i(k,w)}{n}$$

2. Compute the similarity between two nodes u and v by computing the sample standard deviations of all node ratings that they have in common.

$$sim(i,k) = 1 - \sqrt{\frac{\sum_{w \in \text{common}(i,k)} (t(i,w) - t(k,w))^2}{n + m}}$$

B.2 - Definitions

$tr(v,w)$: The amount of trust that node v places in node w .
 n : The number of ratings that node v has made to peers w
 m : The number of ratings that node u has made to peers w
 $sim(u,v)$: The feedback similarity that node u assigns to node v
 $\text{common}(u,v)$: The intersection of the peers that u and v have rated.
 $\text{feed}(u,v)$: The feedback score that peer u assigns to peer v
 $fc(i,j)$: The feedback credibility that peer i assigns to peer j
 $l(i,j)$: The normalized feedback credibility that peer i assigns to peer j
 $c(i,j)$: As defined in B.1
 $\text{edgeweight}(i,j)$: Chance that peer i will propagate its trust to peer j

3. Divide similarity among all nodes m that have a similarity score from node i to create a normalized feedback score.

$$feed(i, k) = \frac{sim(i, k)}{\sum_{i=1}^{R(m)} sim(i, m)}$$

4. Define feedback credibility as a metric that weights each normalized trust score c_{ij} by each normalized feedback score $feed(i, j)$.

$$fc(i, j) = c(i, j) \cdot feed(i, j)$$

5. Redefine local trust value l_{ij} in terms of its normalized counterpart that includes feedback credibility.

$$l(i, j) = \begin{cases} \frac{\max(fc(i, j), 0)}{\sum_{m=1}^{R(i)} \max(fc(i, m), 0)} & \text{if } \sum_{m=1}^{R(i)} \max(fc(i, m), 0) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

6. Redefine initial aggregated trust using the new definition of local trust given above.

$$t(i, j) = \sum_k l(i, k) \cdot c(k, j)$$

7. Set the edgeweight for propagation such that it includes both similarity and normalized trust

$$edgeweight(i, j) = (1 - \beta) \cdot c(j, i) + \beta \cdot sim(j, i)$$

8. [todo]

B.3 - RELATIVE RANK ALGORITHM

1. Separate start-set from non-start-set users

$$A = P'$$

2. Separate non-start set members into groups according to how many k feedbacks each member has received (both positive and negative)

$$A_k \subseteq A \text{ where peers have } k \text{ feedbacks}$$

3. Find the top rated node in every group

$$r(k) = \max(t(i, j)) \in A_k$$

4. Find a line of best fit for all pairs (k, r_k) , and determine the slope m and intercept b of that line

5. Define a non-start set node i 's **relative rank** as an area relative to that line.

$$relativerank(i, r, k) = \frac{r_i - b}{mk}$$

6. Repeat steps 3 - 5, but for set P .

B.3 - Definitions

A: The set of all peers that aren't in the start set P.

P: As defined in B.1

A_k : The subset of A where all peers have k feedbacks.

$t(i, k)$: As defined in B.2

B.4 – VALUE RANK ALGORITHM

1.

B.5 – AFFINITY SCORE ALGORITHM

1. Take the users value rank along all core values for that community, and average them together.

$$affinity(u, c) =$$

B.5 - Definitions

$affinity()$: The function that calculates a user's affinity score
 u : The user who's affinity score is being calculated
 c : The community in which the affinity is being calculated
 A_k : The subset of A where all peers have k feedbacks.

B.6 – TOKEN TRANSFER ALGORITHM

1.

B.7 - TOKEN TRANSFORMATION ALGORITHM

1.

B.8 – PSEUDOSPHERICAL SCORING RULE EQUATION

1.

2. $S_{\beta}^S(r, e_j \| q) = \frac{1}{\beta-1} \left(\left(\frac{r_j/q_j}{(\sum_{i=1}^n (r_i/q_i)^{\beta-1})^{1/\beta}} \right)^{\beta-1} \right) - 1$
3. $\tilde{S}_{\beta}^P(r, e_j \| q) = \sum_{i=1}^{j-1} S_{\beta}^P(R_i, e_2 \| Q_i) + \sum_{i=j}^{n-1} S_{\beta}^P(R_i, e_1 \| Q_i)$

B.8 - Definitions

n : The number of mutually exclusive outcomes that can occur.
 q : A baseline probability distribution that represents initial information
 r : A reported probability distribution
 P : As defined in B.1

B.9 – PSEUDOSPHERICAL INFORMATION GAIN EQUATION

B.10 – POWER SCORING RULE EQUATION

B.11 – POWER INFORMATION GAIN EQUATION

B.12 – CREATIVITY SCORES ALGORITHM

B.13 – CLARITY SCORES ALGORITHM

B.14 – INFLUENCE LIMITER ALGORITHM

B.15 – EXPERT STAKE ALGORITHM

B.16 - CLIENT

APPENDIX C – TECHNICAL DETAILS AND CHALLENGES

EXPENSIVE COMPUTATIONS

Verity has a number of computations that would simply be too expensive to run on-chain. Particularly for Monte-carlo simulations, any decrease in cost and speed creates a corresponding increase in accuracy, so any optimizations made directly impact the quality of the platform. Because of this, it's critical that the costs and speeds of computation are lowered from those typically seen on Ethereum.

On the other hand, the Ethereum blockchain provides availability, accuracy, and immutability guarantees that are critical for an application whose purpose is to quantify trust. It's critical that these promises aren't weakened too severely.

The paradigm that Verity plans to use to balance these two issues is to push expensive computations to off-chain, but to allow complete verification on-chain if there's any disagreements.

This paradigm will be achieved through two emerging technologies: state channels and interactive verification.

STATE CHANNELS

The first technology is state channels. The idea behind state channels is that all parties involved in a transaction must agree on the results of a computation. If they all agree, only the results of the computation are posted on the Blockchain. If any party doesn't agree on the results of the computation, it is then run on-chain to determine whom was correct. An early example of generalized state channels is the concept of "smart state channels" in Raiden.

STATE CHANNELS IN ACTION

One example of where state channels might be used is in determining the payouts for contest rewards and Verity tokens at the conclusion of a contest. If everyone agrees on the proper payouts within some specified time frame, payouts can happen cheaply and instantly, with only a small gas cost needed to update the final distribution of rewards on the blockchain.

If parties do not agree on the proper distribution of tokens, then the payout structure degrades to an on-chain method in which each participant is responsible for claiming their own rewards and therefore updating their own individual state within the smart contract. This method is more costly and time consuming, but its existence ensures that the reputation calculations can always be trusted.

The existence of Verity's value-based reputation disincentivizes "griefing attacks" on the original state-channel based method, as users whom submit wrong answers in order to raise costs for others will lose reputation, and therefore the ability to have financial stake in the community.

INTERACTIVE VERIFICATION

Interactive verification works similarly to state-channels, but with a twist: Firstly, the computations need not be run by people involved in the transactions. Secondly, the full computation need not be done on chain, instead only enough needs to be run to determine which party is correct.

The way this works is that a computation is run completely off-chain by an interested party who posts up a security deposit. The results of this computation can then be challenged by another party, whom also posts up a security deposit. If challenged, the computation is then verified on-chain, and the incorrect party loses their deposit to the correct party. Early inroads to interactive verification were made with Ethereum computation market. More recently, TrueBit presented a method that works even for computations that would be too expensive for Ethereum's gas limit.

INTERACTIVE VERIFICATION IN ACTION

One place in which interactive verification would work for Verity is in its computation of Monte-carlo simulations. Users (especially mobile users) likely won't have the computation power to run the simulations locally, and the full computation will most likely be above Ethereum's gas limit. Both issues rule out state-channels as a viable solution, and thus require an interactive verification approach.

Taking Truebits approach, this would involve submitting a Merkle tree of how one arrived at the final expected value of each contest submission (using a pseudorandom seed provided by the blockchain), as well as the expected value itself to the blockchain. A challenger whom thought the final computation was wrong would then be able to interactively challenge individual parts of the merkle tree where they thought the computation was wrong, and the blockchain would verify who was correct.

GAS COSTS

Gas costs on Ethereum are the costs associated with running computations and storing data on all the nodes in the Ethereum network. While gas costs can be significantly reduced using the methods mentioned in the Expensive Computations section, they can never be completely eliminated.

This presents a challenge for a platform such as Verity that hopes to have broad appeal, as users aren't accustomed to paying for internet platforms, and even small costs have significant overhead in terms of the cognitive load (Szabo, 1999). Thus it remains important to enumerate whom will pay for gas, and how that payment will appear as choices for the user. Verity has separate answers for these questions over the lifetime of the product.

SHORT TERM – MARKET SELECTION AND FOUNDATION SUBSIDIES

In the short term Verity will take two approaches to ensure that gas costs don't prevent adoption of the platform. Firstly, it will make a careful selection of its early markets, specifically targeting users and developers whom understand the technology, and are willing to pay for the benefits of decentralization and transparency. In practice, this means targeting integration with other decentralized applications on Ethereum, whose users and developers are both in this category.

Secondly, the Verity Foundation will work with a select group of partners to integrate with their existing crowdsourcing and sharing economy websites and apps, and completely subsidize the gas usage of these organizations for a number of years. Having immediate integration with the "legacy internet" and the existing "app ecosystem" is incredibly important for the future growth of Verity, and these integrations and subsidies will allow Verity to rapidly reach product-market fit within these key markets.

LONG-TERM – GAS AUTOMATION AND WEBSITE SUBSIDIES

In the mid-term, as the reputation tokens begin to gain real-world value and contests with high rewards become common, Verity will work to create a system of smart-contracts that allows the users to pay for their own gas without conscious interference. By automatically selling tiny fractions of the reputation tokens and rewards that a user earns on a decentralized marketplace, Verity can create a system where users are paying for gas costs without ever having to make the conscious decision to reach into their wallet. A system like this is necessary for widespread user adoption in the long term.

In addition to users paying their own way (which may not work for users whom are performing poorly in contests), websites and apps can subsidize users usage of their website by paying the gas costs themselves, in a similar way that they pay for computations now using a service like Amazon Web Services. At this point, Verity will have a proven use and track record, and can be thought of as a competitive advantage or even a cost-of-doing-business.

RANDOMNESS

One issue with using probabilistic methods like eigenvectors and monte-carlo simulations to determine reputation is that a solid source of randomness is needed that can't be manipulated by users to gain more

reputation. Non-manipulable randomness on the blockchain is hard to come by, and this represents a technical challenge that Verity must solve. To start off, Verity will use a simple semi-trusted oracle for offchain randomness such as OraclizeIt²⁶. Over the long term, Verity would like to mix several sources of randomness together from many different actors with different incentives, such as OracleizeIt, Blockhashes, and a variety of RandAOs²⁷.

²⁶ <http://www.oraclize.it/>

²⁷ <https://github.com/randao/randao>