Crowdsourcing of Formally-Specified Computations

Ahmed Abdelmeged Karl Lieberherr

Northeastern University, CCIS, Boston

mohsen/lieber @ ccs.neu.edu

Abstract

Formally-specified computations are defined by logical statements (a.k.a. claims) interpreted in a computable model. Semantic Games (SGs) of claims provide novel answers to crowdsourcing challenges yet with several limitations. We provide a comprehensive analysis of the limitations of SGs when used for crowdsourcing and propose a new concept, called the Contradiction-Agreement Game (CAG), which builds on SGs and has desirable properties for successful crowdsourcing. The list of desirable properties includes a progress property (each CAG improves evaluation of scholars or adds to social welfare), cheating prevention through anonymity and cheating detection and fairness of the CAG payoff function.

We describe a proof of concept implementation of a CAG-based crowdsourcing platform for formally-specified computational problems, called the Scientific Community Game (SCG). SCG uses a modular construct, called a lab, to group related claims and to solve labs incrementally through lab relations, which are themselves captured as labs. SCG has the flavor of a "Wikipedia for Computations" where inventions are explicit and composable.

Our proposed system has significant applications, in addition to crowdsourcing formally-specified computations. (1) The collaborative and self-evaluating nature of SGs provides a peer-based evaluation system for MOOCs on formal science topics. The peer-based evaluation is guaranteed to be fair, and it saves significant time for the teaching staff. (2) SGs provide a lower barrier of entry to making contributions to formal sciences through game play. It is a significant help to scientists to test claims using the crowd.

1. Background

1.1 Crowdsourcing

Crowdsourcing has become an important problem solving approach that enables us to tackle large scale problems that require human intelligence to solve. Crowdsourcing has been successfully applied to several problems over the past decade. Including, labeling images indexed by Google on the web [39], discovering protein foldings [14], synthesizing proteins [4] and building the Wikipedia.

We are after a "Wikipedia of Computations" where user's inventions are explicit and composable. A computation is specified by an interpreted predicate logic claim in a constructivist setting where computation is performed in some computable model to refute the claim. The claim is refuted when the semantic game associated with the claim is lost by the verifier. The semantic game takes place between the verifier and falsifier of the claim and requires the players to compute objects that defend the players' positions of being verifier or falsifier. For the purpose of this paper we assume that the task of judging claims as well as refuting them can be done by a software. We have avatars playing the role of scholars who recognize true claims and who know how to defend them. Those avatars have a simple interface whose implementation defines an explicit invention which will be judged against other inventions.

Computations in the "Wikipedia of Computations" are queried by a predicate logic claim and either a lab is found which solves that claim or a new lab is defined to initiate a new call to the crowd to solve the claim. Solving the claim means to provide the algorithms necessary to defend it.

What about the quality of the claims in the "Wikipedia of Computations?" That depends on the quality of the crowd participating in a specific lab. If the crowd is strong and interested in the lab, the claims will resist refutation even under heavy attacks. What about spammers trying to infiltrate a lab with false claims? We have three precautions against spammers: (1) they have to defend their claims and (2) they don't know the identity of their adversary (3) they are responsible for the positions they have taken and might have to defend them against other stronger players.

It is our goal to apply crowdsourcing to solve formally-specified computational problems. To achieve this goal, there are four challenging questions that we need to address [16]:

1. What contributions can users make?
2. How to evaluate users and their contributions?
3. How to combine user contributions to solve the target problem?
4. How to recruit and retain users?

1.1.1 Software Development

At first, it appears strange to have a crowd working on the implementation of a formally-specified function (one of the Skolem functions implied by a claim). Why not just hire one good programmer? Isn't the good programmer distracted by having to write an avatar to play semantic games? The formally-specified function can often be implemented in many different ways which have different qualities. To find the optimum quality may be challenging and competition will improve the quality of the software.

It is true that in the end only one good programmer is needed but the problem is how to find her or him. A lab serves this purpose.

The avatar has a meta interface which consists of the following tasks:

1. Choose a claim out of a set of claims.
2. Take a position on a claim, either as verifier (if the avatar thinks the claim is true) or falsifier (if the avatar thinks the claim is false).

3. Play the next move in the semantic game.

Actual avatar interfaces are generated specific to each lab.

Our system uses a Crowd Interaction Mechanism (CIM) to make "optimal" use of the crowd. The CIM tries to create the most information out of the games that are played. For example, if for a claim c there are 2 verifiers v1,v2 and 2 falsifiers f1,f2, the CIM will create 2 games: v1/f1 and v2/f2 and not two games v1/v2 and f1/f2.

Our system might appear expensive because users don't see each others' software (unless there was a "level the boats" event). They have to try to reverse engineer the programs of other users. We feel that this is the price to pay for getting new ideas into the software. We would not recommend our system for formally specified computational problems where no innovation is expected to solve them effectively.

As a running example, consider the $MinBasisSize$ problem:

Size of minimum graph basis: a basis of a directed graph G is defined as set of nodes such that any node in the graph is reachable from some node in the basis. Formally,

$$MinBasisSize(G \in Digraphs, n \in \mathbb{N}) =$$

$$BasisSize(G, n) \wedge \forall k \; s.t. \; k < n : \neg BasisSize(G, k)$$

where

$$BasisSize(G \in Digraphs, n \in \mathbb{N}) =$$
$$\exists s \in \mathcal{P}(nodes(G)) \; s.t. \; |s| = n \; :$$
$$\forall m \in nodes(G) \; \exists p \in paths(G) :$$
$$first(p) \in s \wedge last(p) = m.$$

In the following we assume that the reader is familiar with semantic games. In appendix B there is a concise definition.

The software (avatar) we want to have developed takes a graph G and natural number n and it will tell us whether it wants to be verifier or falsifier. Let's assume, the avatar wants to be verifier and the CIM has found another avatar who wants to be falsifier. Who is right? The semantic game between the two will give us further information:

There are two cases:

- First conjunct

  If the falsifier chooses the first conjunct: $BasisSize(G, n)$, she thinks that there is no vertex basis of size n for G. In this case, the verifier must provide a set s of n nodes and the falsifier must provide a node m and the verifier must provide a path p. If $first(p) \in s \wedge last(p) = m$ the verifier wins, otherwise the falsifier wins.

- Second conjunct

  Remember that negation means that the users switch roles verifier $\iff$ falsifier. If the falsifier chooses the second conjunct: $\forall k \; s.t. \; k < n : \neg BasisSize(G, k)$, she thinks that she can strengthen $MinBasisSize(G, n)$ to a smaller n. In this case, the falsifier chooses a k and a set s of k nodes and the verifier provides a node m and the falsifier provides a path p. If $first(p) \in s \wedge last(p) = m$ the falsifier wins, otherwise the verifier wins.

The above claim $BasisSize(G, n)$ does not discriminate between "hard" and "easy" quantifiers. Indeed, the existential quantifier for s seems harder than the universal quantifier for m and the existential quantifier for p. We can take care of the last two quantifiers with a predicate $reachAll(G, s)$ which checks, using DFS, whether we can reach all nodes in G from s. $reachAll(G, s)$ is implemented in the model of graphs. We now use this function, reducing the communication needs between verifier and falsifier:

$$BasisSize(G \in Digraphs, n \in \mathbb{N}) =$$

$$\exists s \in \mathcal{P}(nodes(G)) \; s.t. \; |s| = n \; : reachAll(G, s)$$

We see in this example that the lab designer has a choice how the avatars (or users, in general) are engaged. Notice that the reduction of quantifiers simplifies the semantic game.

We want to solve the $MinBasisSize(G, n)$ problem incrementally. Let's first make the simplifying assumption that G is a DAG (directed acyclic graph). It seems easier to solve the $MinBasisSize$ problem for this special class of graphs. We have now a new lab where the claims are of the form:

$$BasisSize(G \in DAGs, n \in \mathbb{N}) =$$

$$\exists s \in \mathcal{P}(nodes(G)) \; s.t. \; |s| = n \; : reachAll(G, s).$$

Let's assume we have a solution for this lab. We are lucky to have now a solution for the general graphs because there exists a mapping from general graphs to DAGs that preserves reachability. A path in the original graph translates into a path (possibly of length 0) in the DAG. This results by itself in a lab for the transformation. We have a graph $G_1 = (V_1, E_1)$ and we construct a new DAG $G_2 = (V_2, E_2)$ from G_1. We claim there exists a mapping $f(G_1) \rightarrow G_2$ with two important properties: (1) the **defense (refutation) of claim** $BasisSize(G_2, n)$ **results in a defense (refutation) of claim** $BasisSize(G_1, n)$. (2) there is no information loss: $\forall G_1$: if $BasisSize(G_1, n)$ then $BasisSize(f(G_1), n)$.

1.2 Logical Games and Computational Problems

Logical games have a long history going back to Socrates. More recently, they became a familiar tool in many branches of logic. Important examples are Semantic Games (SGs) used to define truth, back-and-forth games used to compare structures, and dialogue games to express (and perhaps explain) formal proofs [30], [18], [23].

SGs are played between two players, the *verifier* and the *falsifier* [1]. An instructive way of viewing SGs is in their extensive form, which essentially is a tree structure with the root labeled by the formula ϕ , the subsequent labeled nodes representing the subformulas of ϕ, and the vertices labeled by the actions of the players.

In the theory of SGs, logical statements interpreted in a computable model (a.k.a. claims) derive their meaning from the games played by the rules prompted by the logical connectives encountered in the claims [32]. The existence of a winning strategy for the *verifier* implies that the underlying logical statement is indeed *true* and the existence of a winning strategy for the *falsifier* implies that the underlying logical statement is indeed *false*.

Players need to solve *computational problems* in the course of playing SGs. For example, the falsifier of the $prime(7) = \forall k \; s.t. \; 1 < k < 7 : \neg divides(k, 7)$ needs to compute the factors of 7. Similarly, claims can be used to logically specify computational problems. For example, consider the problem of finding the factors of a given natural number $factors(n)$. This problem can be logically specified using the claim $\forall n \exists s : \forall k : divides(k, n) \Leftrightarrow k \in s$. In SGs derived from this claim, the verifier needs to correctly solve $factors(n)$ in order to win.

A computational problem can be logically specified as a claim about the relation between either (1) the input properties and the output properties, or (2) the input properties and the output finding process properties such as resource consumption.

2. Thesis

Our thesis is that semantic games of interpreted logic statements provide a useful foundation for building *successful* crowdsourcing systems for solving computational problems.

[1] Other names has been also used in the literature such as *I* and *Nature*, *Proponent* and *Opponent*, and *Alice* (female) and *Bob* (male).

2.1 Rationale and Limitations of Semantic Games

SGs of claims provide attractive answers to the four challenging questions of crowdsourcing systems. However, these answers are only valid in a limited context. A *successful* SG-based system must generalize SGs to a much wider context and improve on the way SGs address these four challenging questions, whenever possible.

An example of such limitation is that SGs define an interaction mechanism between two users only. A successful SG-based crowdsourcing system must provide a Crowd Interaction Mechanism (CIM) on top of SGs that decides which SGs to be played. To decide on a game to be played, the CIM must decide on a claim, a user to take on the verifier position and a user to take on the falsifier position. On one hand it is important that the CIM relies on user preferences to enhance the user experience, ensure that user contributions are potentially correct, and to ensure the fairness of SG-based evaluation. On the other hand, the overall system would be ineffective if the CIM was just a proxy to users' preferences because the CIM would no longer be able to *drive* the user interaction. For example, it would be impossible to hold an SG between two arbitrary users unless they hold contradictory positions on the same claim.

2.1.1 User Contributions

During the course of playing an SG, users make two kinds of *formal* contributions: positions and supporting actions. These two kinds of contributions can be extracted from SG traces as follows:

The trace of an SG can be represented as a *directed line graph* where nodes represent the state of the SG and edges represent transitions. The state is a tuple consisting of a claim and a pair of players, the player taking the verifier position and the player taking the falsifier position. For example, the tuple $\langle c, p_1, p_2 \rangle$ represents a state where p_1 is the player taking the verifier position and p_2 is the player taking the falsifier position on claim c. A labeled transition represents a supporting action while an unlabeled transition represents an implied action. Implied actions are automatically carried out by the system. An example of implied actions is given by: $\langle \neg c, p_1, p_2 \rangle \rightarrow \langle c, p_2, p_1 \rangle$. Supporting actions are either attacks or defenses, and they involve an additional parameter that one of the users must provide. For example, the transition $\langle \forall x : p(x), p_1, p_2 \rangle \xrightarrow{x_0} \langle p(x_0), p_1, p_2 \rangle$ is an attack made by p_2, where x_0 is a counter example provided by p_2.

Apart from playing SGs, users can still contribute by improving their own SG playing strategies. Players, by improving their strategies, are able to spot more problems in the positions taken by their opponents in future games. Because users have to follow a well defined formal protocol B to play an SG, this enables users to *automate* the execution of their strategies into *avatars*. Algorithms used in avatars are themselves yet another potential formal contribution (see Section 2.2).

2.1.2 Evaluating Users

SGs provide an *objective* and *self-sufficient* approach to assess the *relative strength* of users. Simply put, the winner of an SG is considered *stronger* than the loser. This approach is fundamentally different from the current evaluation schemes used in crowdsourcing systems such as: gold standards, trusted workers and probabilistic oracles, and disagreement-based schemes [21].

Disagreement-based schemes evaluate the *absolute strength* of users based on how often the user's contribution is "correct" where a "Correct" contribution is defined to be *similar* to the "majority vote". SG-based evaluation is independent of the "correctness" of user contributions. Instead SG-based evaluation can *objectively* judge one contribution to be "better" than the other. It is worth noting that the "better" contribution is not always necessarily *similar* to the "majority vote".

SG-based evaluation is said to be *self-sufficient* because, unlike gold standard evaluation, it is not based on a set of pre-populated test cases. Instead, the two users test each other.

It is important to evaluate users' strength based on their performance in a large number of SGs. The naïve approach of summing the number of SGs the user won is unlikely to be fair due to several concerns that give one group of players an advantage over another group of players. A comprehensive list of these concerns is given by:

1. Users can be at an advantage (or at a disadvantage) if they participate in more SGs where they are at an advantage (or at a disadvantage). A player is at an advantage (or at a disadvantage) in an SG if either the claim **(CONCERN 1.a)** or the position **(CONCERN 1.b)** is only forced on their adversary (or only forced on them).
2. Users can be at an advantage (or at a disadvantage) if they participate in more (or fewer) than the average number of SGs played by their counterparts **(CONCERN 2)**.
3. Users can be at an advantage (or at a disadvantage) if they participate in more SGs against other weaker (or stronger) users **(CONCERN 3)**.
4. If a group of users can form a coalition with the goal of artificially increasing the strength of a particular user through losing against that user on purpose, then the winning user is at an advantage **(CONCERN 4)**.

As we mentioned before, it would not be effective to address the first concern by ensuring that, in every game, neither of the players is at an advantage (or a disadvantage). Instead, the system has to adopt a non-local view on fairness and ensure that none of the players in the crowd is at an advantage (or a disadvantage) considering all played SGs. The second and third concerns can be addressed through either restricting the algorithm by which the system decides which SGs to be played, or through a more sophisticated approach to assess the user strength, or through both approaches. Anonymity can be used to defend against the fourth concern.

2.1.3 Evaluating User Contributions

Based on the outcome of an SG, we cannot safely assume that certain contributions are "correct". Therefore, the best we can do is to judge certain user contributions to be *potentially correct*. We consider contributions to be potentially correct if we have no reason to believe they are potentially incorrect.

By definition, the contributions of an SG loser are "Incorrect". Other reasons to believe that certain contributions are potentially incorrect include:

1. The position taken by the winner was forced **(CONCERN 5)**.
2. There is no mechanism to discourage "cheating" (i.e. *knowingly* making "incorrect" contributions) either because their adversary is weak enough not to discover the "cheat", or to lose on purpose against their opponent **(CONCERN 6)**.

Anonymity can be used to discourage "cheating". It is also possible to hold the positions taken by users against themselves in future SGs.

2.1.4 Combining User Contributions

It is possible to collect the potentially correct contributions of all winners of SGs into a contribution database. The *crowd beliefs* about claims can be assessed from the contribution database. It is possible that "incorrect" contributions make it to the contribution database **(CONCERN 7)**. Therefore, it is necessary to have a

periodic mechanism to clean the contribution database in order to enable more accurate assessment of the crowd beliefs.

Apart from estimating the crowd beliefs, SG losers get precise feedback on how they can improve their SG playing strategies. Furthermore, users can then build on the crowd beliefs. For example, suppose that the winners were mostly taking the verifier position on the claim $\forall k : divides(k, 3571) \Leftrightarrow k \in \{1, 3571\}$, then this likely-to-be-true claim can be used as a test case for factorization algorithms.

2.1.5 Recruiting and Retaining Users

Participating in an SG can provide users with an intrinsically rewarding experience. The exact intrinsic rewarding experience is user dependent. For example, some participants can find the act of game play against an adversary to be fun. Others can enjoy the educational (or collaborative) nature of SGs that comes from the fact that the winner of an SG gives the loser very targeted feedback.

We believe that the following three factors could enhance the intrinsically rewarding experience that SGs provide to users:

1. Choosing claims that both players find interesting **(CONCERN 8)**.
2. Allowing users to choose their positions on claims **(CONCERN 9)**.
3. Matching players with similar levels of strength **(CONCERN 10)**.

Neither intrinsic nor extrinsic reward is absolutely superior [2] [3]. However, most certainly, a crowd would have users that prefer both kinds of rewards. Therefore, it is still useful to include other encouragement and retention schemes **(CONCERN 11)** such as instant gratification, providing ways to establish, measure, and show different qualities of the users, establishing competitions and providing ownership situations [16].

2.2 Applications

In this paper we use an SG-based system for crowdsourcing computational problem solving. However, there are several other significant applications to teaching, crowdsourcing software development and crowdsourcing formal science.

2.2.1 Teaching

The collaborative and self-evaluating nature of SGs is useful in teaching (especially MOOCs) where teaching other students helps boost one's evaluation. The winner against a non-forced opponent teaches the opponent a lesson.

2.2.2 Software Development

The mandatory use of formal specification of claims and the orderly nature of the semantic games enables the system to be used as a crowdsourcing system for algorithms for computational problems as well. Because users can "automate themselves" as avatars (programs). The strongest avatars would have good algorithms either for generating tests for other avatars or solving a computational problem or both.

[2] For example, consider using Amazon Mechanical Turk (AMT) to label all images indexed by Google. Would that be as cost effective as the ESP game? A second example is building the Wikipedia. Would it be as cost effective to build the Wikipedia using AMT?

[3] Extrinsic reward is believed to be superior in motivating automatic (motor) tasks, while intrinsic value would be superior in motivating intelligent (cognitive) tasks [33], [22], [19].

2.2.3 Formal Science

Although scientists in formal sciences are often interested in finding proofs to their claims, it remains helpful to test those claims first with the help of the crowd. Testing can provide them with useful insights. For example, testing can reveal a corner case where the claim does not hold. Reformulating the original claim to avoid such corner cases could be helpful in finding proofs [3]. It is worth mentioning that the phrase "formal science" is not limited to mathematics and logic. It also applies to scientific uses of formal simulation models.

3. Initial Investigation

To support our thesis, we designed and partially implemented [1] a proof of concept SG-based crowdsourcing system. Our system constitutes a redesign from scratch of the Scientific Community Game (SCG) [7], [6], [29] which has been evolving since 2007. Below, we describe our newly designed system and report on our experience of using earlier iterations of SCG for teaching.

3.1 System Overview

In a nutshell, our system uses first-order logic to express claim families (See Appendix A for more details), and uses the semantic games of first-order logic formulas defined by Hintikka's Game-Theoretic-Semantics [25] (See Appendix B for more details).

To ensure that claims are never forced on users, our system uses labs. Labs define special interest groups of users. A lab is created by an owner (one kind of users) and consists of a family of claims. Scholars (another kind of user) choose to join the labs they find *interesting*. The system only allocates users to SGs of claims from the labs they joined. This enhances the users' experience while participating in SGs **(CONCERN 8)** and guarantees that users are never at a disadvantage regardless of the method used to chose the underlying claims for SGs **(CONCERN 1.a)**.

Rather than making scholars participate in SGs directly, the CIM in our system makes users participate in Contradiction-Agreement Games (CAGs). Although CAGs are composed of SGs, CAGs can be played by two players taking the same position on the underlying claims. This enhances the users' experience **(CONCERN 9)**. Furthermore, CAGs are specifically designed to provide a fair evaluation **(CONCERN 1.b)** and to identify potentially correct contributions **(CONCERN 5)**. CAGs are described in Section 3.2. Currently, our system has a per-lab CIM. Lab owners are required to provide their CIM mechanisms, e.g., to match scholars with close enough strength. This is critical to enhance the users' experience **(CONCERN 10)** and fairness **(CONCERN 3)**.

Our system uses an algorithm to evaluate the users' strength as fairly as possible. Our algorithm is designed to address the fairness concerns **(CONCERN 2,3)**. The algorithm is described in Section 3.3. To estimate crowd beliefs, our system uses a simple formula that is presented in Section 3.4. To discourage "cheating" **(CONCERN 4,6)**, our system relies on anonymity. Currently, our system does not provide a mechanism for cleaning the contributions database **(CONCERN 7)** nor any encouragement and retention schemes **(CONCERN 11)** other than the fun that scholars get from participating in SGs.

3.2 The Contradiction-Agreement Game

CAGs remove the restriction that scholars must take contradictory positions on claims. In case scholars take contradictory positions, CAG reduces to one SG. Otherwise, CAG reduces to two testing SGs. In a test SG, one of the scholars, the tester, is forced to take the opposite position of the position it chose. The two scholars switch their testing roles between the two games. Even though, the tester is forced to take a particular position, CAG-based evaluation remains

Game	forced	winner	payoff (p_1, p_2)	potentially correct contribution
Agreement T1	p_2	p_1	(0, 0)	p_1
	p_2	p_2	(0, 1)	–
Agreement T2	p_1	p_1	(1, 0)	–
	p_1	p_2	(0, 0)	p_2
Contradiction	–	p_1	(1, 0)	p_1
	–	p_2	(0, 1)	p_2

Table 1. The Contradiction-Agreement Game

fair. It also remains possible to get potentially correct contributions out of the testing games when the winner is not the forced tester.

SGs with forced scholars can cause unfairness in two different ways:

1. Winning against a forced scholar is not the same as winning against an unforced scholar. Giving both winners a point for winning would be unfair.
2. The forced scholar is at a disadvantage.

To overcome these two problems, we adopt the rule that the scholar winning an SG *scores* a point only if its adversary is not forced. Although, this solves the two problems, it, oddly enough, puts the winner at a disadvantage because it has no chance of *scoring* a point. Luckily, considering both test games together, the evaluation (i.e. payoff) is fair because both scholars have an equal chance of scoring. Furthermore, scholars remain properly incentivised to win under the payoff. This is important to ensure the fairness of user evaluation as well as the potential correctness of the contributions of the unforced winners. Our readers can verify these properties by inspecting Table 1 which summarizes CAGs. The columns of the table indicate the name of the SG being played, the forced scholar (if any), the SG winner, and whether the contribution of the winner is potentially correct (assuming that "cheating" is somehow discouraged).

3.2.1 CAG Desirable Properties

CAG encourages innovation because forced scholars can score while their adversary cannot. This provides an incentive for forced players to win SGs even though they are forced to take positions that are often contradictory to their own intuition as well as to the crowd beliefs. Also, CAGs ensure that some form of progress is taking place either as an update to the player scores or that a potentially correct contribution has been made. Furthermore, in the first case, the loser is receiving targeted feedback and in the second case, the community benefits from the potentially correct contribution.

3.3 Evaluating User Strength

We devised an algorithm to evaluate user strength based on CAG scores. The algorithm weighs the scores by the strength of the adversary and calculates the strength of the scholar as the ratio of wins over the sum of wins and losses in order to even out the difference in the number of played CAGs **(CONCERN 2)** as well as the difference in the strength of adversaries **(CONCERN 3)**.

Informally, the algorithm starts with an estimate of 1 for the strength of all players. Then it computes the weighted wins and losses for each player based on the payoffs and the strength of their adversaries. Then it computes strength as the fraction of weighted wins divided by the sum of weighted wins and losses. The last two steps are iterated to a fixpoint.

Formally, we denote the sum of payoffs that scholar S_1 gets from scholar S_2 by $\mathit{Payoff}(S_1, S_2)$. The strength of user S is denoted by $Str(S)$. The algorithm is given by:

$$
\begin{aligned}
Str^{(-1)}(S_i) &= 1 \\
Wins^{(k)}(S_i) &= \sum \mathit{Payoff}(S_i, S_j) * Str^{(k-1)}(S_j) \\
Losses^{(k)}(S_i) &= \sum \mathit{Payoff}(S_j, S_i) * (1 - Str^{(k-1)}(S_j)) \\
Total^{(k)}(S_i) &= Wins^{(k)}(S_i) + Losses^{(k)}(S_i) \\
Str^{(k)}(S_i) &= \begin{cases} Wins^{(k)}(S_i)/Total^{(k)}(S_i), & \text{if } Total^{(k)} \neq 0 \\ 0.5, & \text{otherwise.} \end{cases}
\end{aligned}
$$

Ideally, we would like the strengths produced by the algorithm to be *consistent* with the payoffs (i.e. $\forall S_1, S_2 : \mathit{Payoff}(S_1, S_2) \geq \mathit{Payoff}(S_2, S_1) \Rightarrow Str(S_1) \geq Str(S_2)$). However, the relation $R(S_1, S_2) = \mathit{Payoff}(S_1, S_2) \geq \mathit{Payoff}(S_2, S_1)$ is not necessarily transitive while the relation $Q(S_1, S_2) = Str(S_1) \geq Str(S_2)$ is. However, we conjecture that the strengths produced by our algorithm minimize such inconsistencies.

However, the the algorithm possesses the following weaker soundness properties:

1. A scholar S_i that beats the score of another scholar S_j on their mutual games as well as on games with all other scholars S_k will have a higher strength. $\forall i, j \mathit{Payoff}(S_i, S_j) > \mathit{Payoff}(S_j, S_i) \wedge \forall k \neq i, j : \mathit{Payoff}(S_i, S_k) \geq \mathit{Payoff}(S_J, S_k) \wedge \mathit{Payoff}(S_j, S_k) \leq \mathit{Payoff}(S_i, S_k) \Rightarrow Str(S_i) \geq Str(S_j)$.
2. A scholar that only won(lost) games will have a strength of 1(0). Formally, $\forall i \forall j \mathit{Payoff}(S_i, S_j) = 0 \wedge \exists j \mathit{Payoff}(S_j, S_i) > 0 \Rightarrow Str(S_i) = 0$, and $\forall i \forall j \mathit{Payoff}(S_j, S_i) = 0 \wedge \exists j \mathit{Payoff}(S_i, S_j) > 0 \Rightarrow Str(S_i) = 1$. A scholar that has not won or lost any games will have a strength of 0.5. Formally, $\forall i \forall j \mathit{Payoff}(S_j, S_i) = 0 \wedge \mathit{Payoff}(S_i, S_j) = 0 \Rightarrow Str(S_i) = 0.5$.

3.4 Evaluating Crowd Beliefs

We consider the positions taken by non-forced CAG winners to be providing the community with an evidence that these positions are correct. We take the strength of the losing user as the weight of such evidence. For each claim c we let c_T be the sum of the weights of all evidences that c is true, c_F be the sum of the weights of all evidences that c is false. The believed likelihood that c is true is $C_T/(C_T + C_F)$. Similarly, the believed likelihood that c is false is $C_F/(C_T + C_F)$.

4. Experience with the SCG

The SCG has evolved since 2007. We have used the SCG in software development courses at both the undergraduate and graduate level and in several algorithm courses. Detailed information about those courses is available from the second author's teaching page.

4.1 Software Development

The most successful graduate classes were the ones that developed and maintained the software for SCG Court [5] as well as several labs and their avatars to test SCG Court. Developing labs for avatars has the flavor of defining a virtual world for artificial creatures. At the same time, the students got detailed knowledge of some problem domain and how to solve it. A fun lab was the Highest Safe Rung lab from [24] where the best avatars needed to solve a constrained search problem using a modified Pascal triangle.

4.2 Algorithms

The most successful course (using [24] as textbook) was in Spring 2012 where the interaction through the SCG encouraged the students to solve difficult problems. Almost all homework problems were defined through labs and the students posted both their ex-

ploratory and performatory actions on piazza.com. We used a multiplayer version of the SCG binary game which created a bit of an information overload. Sticking to binary games would have been better but requires splitting the students into pairs. The informal use of the SCG through Piazza (piazza.com) proved successful. All actions were expressed in JSON which allowed the students to use a wide variety of programming languages to implement their algorithms.

The students collaboratively solved several problems such as the problem of finding the worst-case inputs for the Gale-Shapley algorithm (see the section Example above).

We do not believe that, without the SCG, the students would have created the same impressive results. The SCG effectively focuses the scientific discourse on the problem to be solved.

The SCG proved to be adaptive to the skills of the students. A few good students in a class become effective teachers for the rest thanks to the SCG mechanism.

5. Related Work

5.1 Crowdsourcing and Human Computation

There are several websites that organize competitions. What is common to many of those competitions? We believe that the SCG provides a foundation to websites such as TopCoder.com or kaggle.com.

The SCG makes a specific, but incomplete proposal of a programming interface to work with the global brain [11]. What is currently missing is a payment mechanism for scholars and an algorithm to split workers into pairs based on their background.

The SCG is a generic version of the "Beat the Machine" approach for improving the performance of machine learning systems [10].

Scientific discovery games, such as FoldIt and EteRNA, are variants of the SCG. [13] describes the challenges behind developing scientific discovery games. [9] argues that complex games such as FoldIt benefit from tutorials. This also applies to the SCG, but a big part of the tutorial is reusable across scientific disciplines.

5.2 Logic and Imperfect Information Games

Logic has long promoted the view that finding a proof for a claim is the same as finding a defense strategy for a claim.

Logical Games [30], [18] have a long history going back to Socrates. The SCG is an imperfect information game which builds on Paul Lorenzen's dialogical games [23].

5.3 Foundations of Digital Games

A functioning game should be deep, fair and interesting which requires careful and time-consuming balancing. [20] describes techniques used for balancing that complement the expensive playtesting. This research is relevant to SCG lab design. For example, if there is an easy way to refute claims without doing the hard work, the lab is unbalanced.

5.4 Architecting Socio-Technical Ecosystems

This area has been studied by James Herbsleb and the Center on Architecting Socio-Technical Ecosystems (COASTE) at CMU http://www.coaste.org/. A socio-technical ecosystem supports straightforward integration of contributions from many participants and allows easy configuration.

The SCG has this property and provides a specific architecture for building knowledge bases in (formal) sciences. Collaboration between scholars is achieved through the scientific discourse which exchanges instances and solutions. The structure of those instances and solutions gives hints about the solution approach. An interesting question is why this indirect communication approach works.

The NSF workshop report [35] discusses socio-technical innovation through future games and virtual worlds. The SCG is mentioned as an approach to make the scientific method in the spirit of Karl Popper available to CGVW (Computer Games and Virtual Worlds).

5.5 Online Judges

An online judge is an online system to test programs in programming contests. A recent entry is [31] where private inputs are used to test the programs. Topcoder.com includes an online judge capability, but where the inputs are provided by competitors. This dynamic benchmark capability is also expressible with the SCG: The claims say that for a given program, all inputs create the correct output. A refutation is an input which creates the wrong result.

5.6 Educational Games

The SCG can be used as an educational game. One way to create adaptivity for learning is to create an avatar that gradually poses harder claims and instances. Another way is to pair the learner with another learner who is stronger. [8] uses concept maps to guide the learning. Concept maps are important during lab design: they describe the concepts that need to be mastered by the students for succeeding in the game.

5.7 Formal Sciences and Karl Popper

James Franklin points out in [17] that there are also experiments in the formal sciences. One of them is the 'numerical experiment' which is used when the mathematical model is hard to solve. For example, the Riemann Hypothesis and other conjectures have resisted proof and are studied by collecting numerical evidence by computer. In the SCG experiments are performed when the refutation protocol is elaborated.

Karl Popper's work on falsification [34] is the father of non-deductive methods in science. The SCG is a way of doing science on the web according to Karl Popper.

5.8 Scientific Method in CS

Peter Denning defines CS as the science of information processes and their interactions with the world [15]. The SCG makes the scientific method easily accessible by expressing the hypotheses as claims. Robert Sedgewick in [36] stresses the importance of the scientific method in understanding program behavior. With the SCG, we can define labs that explore the fastest practical algorithms for a specific algorithmic problem.

5.9 Games and Learning

Kevin Zollman studies the proper arrangement of communities of learners in his dissertation on network epistemology [40]. He studies the effect of social structure on the reliability of learners.

In the study of learning and games the focus has been on learning known, but hidden facts. The SCG is about learning unknown facts, namely new constructions.

5.10 Origins of SCG

A preliminary definition of the SCG was given in a keynote paper [29]. [26] gives further information on the . The original motivation for the SCG came from the two papers with Ernst Specker: [27] and the follow-on paper [28]. Renaissance competitions are another motivation: the public problem solving duel between Fior and Tartaglia, about 1535, can easily be expressed with the SCG protocol language.

6. Conclusion and Future work

We presented SCG, a crowdsourcing platform for computational problems. SCG provides a simple interface to a community that

uses the (Popperian) Scientific Method. Our future work includes further development to the current system, its underlying model, as well as to evaluate our system.

6.1 Model Development

6.1.1 Claim Family Relations and Meta Labs

Relations computational problems can be used to *test* implementations of their solution algorithms. Reduction is an important kind of relation between computational problems that can be used to *prove* certain impossibility results as well as to enable the implementation of one computational problem to *reuse* the implementation of another computational problem.

In our system, these relations can be expressed as claims between claim families specifying computational problems. For example, consider the following two claim families:

1. *Size of minimum graph basis*: a basis of a directed graph G is defined as set of nodes such that any node in the graph is reachable from some node in the basis. Formally, $MinBasisSize(G \in Digraphs, n \in \mathbb{N}) = BasisSize(G, n) \wedge \forall k \; s.t. \; k < n : \neg BasisSize(G, k)$ where $BasisSize(G \in Digraphs, n \in \mathbb{N}) = \exists s \in \mathcal{P}(nodes(G)) \; s.t. \; |s| = n : \forall m \in nodes(G) \; \exists p \in paths(G) : first(p) = m \wedge last(p) \in s$.
2. *Number of source nodes of a DAG*: a source node is a node with no incoming edges. Formally, $\#src(D \in DAGs, m \in \mathbb{N}) = \exists s \in \mathcal{P}(nodes(D)) \; s.t. \; |s| = m \; : \forall v \in nodes(D) : inDegree(v) = 0 \Leftrightarrow v \in s$.

The relation between the two claim families $MinBasisSize(G \in Digraphs, n \in \mathbb{N})$ and $\#src(D \in DAGs, m \in \mathbb{N})$ can be described by $\forall G \in Digraphs, n \in \mathbb{N} \; : MinBasisSize(G, n) = \#src(SCCG(G), n)$ where $SCCG$ refers to Tarjan's the Strongly Connected Component Graph algorithm.

Like other claims, claims about relations between claim families can be studied in a regular lab in our system. However, we see a potential to further utilize these claims to cross check crowd beliefs across labs and to translate user contributions across labs. To harness this potential, we propose to add meta labs to our system. More specifically, we propose to answer the two following questions:

1. How can our system further utilize relations between claim families beyond regular claims?
2. How to express meta labs so that it is possible to further utilize them in an automated way?

6.1.2 Generalized Claims

Users can lose SGs involving optimization problems even if their solutions can be *almost* optimal. For example, consider the following claim: $\forall p \in Problem : \exists s \in Solution : \forall t \in Solution : better(quality(s, p), quality(t, p))$ It is enough for the falsifier to provide a *slightly* better solution to win. As remedy, it is possible to bias the situation towards the verifier by requiring the falsifier to provide at solution that is at well better than the solution provided by the verifier in order to win. The following claim illustrates this solution: $\forall p \in Problem : \exists s \in Solution : \forall t \in Solution : within10\%(quality(s, p), quality(t, p))$ It is also possible to generalize the claims such that the larger the quality gap is the more of a payoff the winner gets. For example, assuming the $distance$ function returns a number between -1 and 1, the following generalized claim illustrates this solution: $\forall p \in Problem : \exists s \in Solution : \forall t \in Solution : distance(quality(s, p), quality(t, p))$ We propose to develop a systematic approach for computing the payoff in SGs for generalized claims.

6.2 System Development

We propose to turn the current implementation [1] into a web based application. We also propose to further develop the claim language and the CIM and the encouragement and retention scheme.

6.2.1 Claim Language

There are certain game related concerns that cannot be expressed in the current claim language. Furthermore, the current language is not as user friendly as it could be. To overcome these two problems, we propose to make the following enhancements to the claim language:

1. make the claim language support second order logical sentences. This enables the claim language to express properties about the resource consumption of algorithms. For example, $AlgoRunTime(c, n_{min}, n_{max}) = \exists a \in Algo : \forall i \in Input \; s.t. \; n_{min} \leq size(i) \leq n_{max} : correct(i, a(i)) \wedge RunTime(a(i)) <= c * size(i)$.
   Second order logical sentences can also express the dependence (or independence) of atoms through Skolem functions. Other approaches to express the dependence concerns is through either the dependence friendly logic or the independence friendly logic [38].
2. add a let binder for efficiency. For example, to avoid computing $a(i)$ twice in $AlgoRuntime$.
3. add syntactic forms, in addition to $Formula$, to provide a more user friendly support of different kinds of computational problems (such as search, optimization, counting problems). For example, to enable users to write: $sat(f) = \max_J csat(f, J)$, instead of: $sat(f, x) = \exists J \; s.t. \; csat(f, J) = x : \forall H : csat(f, H) \leq csat(f, J)$
4. add an abstraction facility.

6.2.2 Crowd Interaction Mechanism

We propose to implement the following CIM. Lab owners establish Swiss-style CAG tournaments between scholars in the lab in order to drive interactions in the lab. Swiss-style tournaments have the property of matching players with similar strength and therefore enhancing the users' experience **(CONCERN 10)** as well as fairness **(CONCERN 3)**. Claims can be chosen by one of the following approaches:

1. *CAG matches*: CAG matches consist of an even number of CAGs. Each scholar chooses the claim for exactly half of the CAGs. Claims must be chosen from the lab's claim family. A generalization of this approach is to play a cut-and-choose game [18] where in each round, one players chooses a set of claims (a cut) then the adversary chooses a claim from the set.
2. *Owner dictated*: the lab owner provides an algorithm for selecting claims to achieve a particular purpose. For example, if the purpose is to clean the contribution database, then the algorithm would select claims underlying scholar contributions in the contribution database. The purpose could also to solve a particular subset of open problems or to solve a computational problem in a particular approach, delegating subproblems to the crowd. For example, the purpose could be to *plot* the relationship between a particular claim family parameter and the correctness of the claim.
3. *Battleship style*: use a claim that both scholars had previously contributed a position on.

A distinctive feature of this CIM is that scholars never choose their adversaries. This is important to discourage "cheating" **(CONCERN 4,6)**. A second feature is that CIM *memoizes* winning po-

sitions taken by scholars and never asks scholars to provide these positions in subsequent CAGs until scholars fail to defend these positions. The contribution database plays the role of the cache for winning positions. This is important to discourage "cheating" **(CONCERN 6)**. It is also important that the CIM allows scholars to revise their previously established contributions to avoid losing future CAGs

6.2.3 Encouragement and Retention Scheme

We suppose that scholars will aspire to have the highest scores on meaningful performance measures. The system can establish the scores for players and provide few different views, such as a leaderboard, for scholars to encourage score based competition. In addition to strength, we propose to develop the following complementary measures:

1. *Breakthrough contribution* : When required to do so, scholars might provide well known claims that they know how to defend. The purpose of developing a measure for breakthrough contributions is to encourage scholars to propose new claims and take positions opposing to the crowd beliefs.
2. *Learning* : Learning is an indirect contribution of scholars. We propose to assess learning through the change in scholar's strength as well as through scholar's revisions to its own established contributions.
3. *Crowd preference* : Scholars might be able to spot certain attractive properties of a particular contribution. The idea is to enable scholars to "like" contributions and essentially count the "likes" the contributions of a particular scholar gets.

Another potential encouragement and retention scheme that we want to explore is to have an underlying theme where scholars are represented by customizable virtual avatars. This enhances the engagement as scholars can become invested in customizing their avatars besides it makes it easier for scholars to be embodied in CAGs by their avatar.

6.3 Evaluation

We propose to conduct a two part evaluation of effectiveness of our system in leveraging the problem solving ability of the crowd. The first part consists of evaluating the quality of the algorithms produced by the crowd to solve non-trivial computational problems. We propose to compare those algorithms to the best known algorithms. Examples include the max cut problem and the highest safe rung problems. The second part consists of comparing the quality of the algorithms produced by the crowd through our system to algorithms produced through traditional crowdsourcing competitions. Examples include the genome-sequencing-problem [2].

We propose to also use our crowdsourcing system to evaluate a number of its components and their properties.

References

[1] Website. `https://github.com/amohsen/fscp`.

[2] Algorithm development through crowdsourcing. `http://catalyst.harvard.edu/services/crowdsourcing/algosample.html`.

[3] The polymath blog. Website. `http://polymathprojects.org/`.

[4] EteRNA. Website, 2011. `http://eterna.cmu.edu/`.

[5] A. Abdelmeged and K. J. Lieberherr. SCG Court: Generator of teaching/innovation labs on the web. Website, 2011. `http://sourceforge.net/p/generic-scg/code-0/110/tree/GenericSCG/`.

[6] A. Abdelmeged and K. J. Lieberherr. The Scientific Community Game. In *CCIS Technical Report NU-CCIS-2012-19*, October 2012. `http://www.ccs.neu.edu/home/lieber/papers/SCG-definition/SCG-definition-NU-CCIS-2012.pdf`.

[7] A. Abdelmeged and K. J. Lieberherr. FSCP: A Platform for Crowdsourcing Formal Science. In *CCIS Technical Report*, February 2013. `http://www.ccs.neu.edu/home/lieber/papers/SCG-crowdsourcing/websci2013_submission_FSCP.pdf`.

[8] E. Andersen. Optimizing adaptivity in educational games. In *Proceedings of the International Conference on the Foundations of Digital Games*, FDG '12, pages 279–281, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1333-9. doi: 10.1145/2282338.2282398. URL `http://doi.acm.org/10.1145/2282338.2282398`.

[9] E. Andersen, E. O'Rourke, Y.-E. Liu, R. Snider, J. Lowdermilk, D. Truong, S. Cooper, and Z. Popovic. The impact of tutorials on games of varying complexity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 59–68, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1015-4. doi: 10.1145/2207676.2207687. URL `http://doi.acm.org/10.1145/2207676.2207687`.

[10] J. Attenberg, P. Ipeirotis, and F. Provost. Beat the machine: Challenging workers to find the unknown unknowns. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[11] A. Bernstein, M. Klein, and T. W. Malone. Programming the global brain. *Commun. ACM*, 55(5):41–43, May 2012. ISSN 0001-0782. doi: 10.1145/2160718.2160731. URL `http://doi.acm.org/10.1145/2160718.2160731`.

[12] B. Chadwick. DemeterF: The functional adaptive programming library. Website, 2008. `http://www.ccs.neu.edu/home/chadwick/demeterf/`.

[13] S. Cooper, A. Treuille, J. Barbero, A. Leaver-Fay, K. Tuite, F. Khatib, A. C. Snyder, M. Beenen, D. Salesin, D. Baker, and Z. Popović. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 40–47, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-937-4. doi: 10.1145/1822348.1822354. URL `http://doi.acm.org/10.1145/1822348.1822354`.

[14] S. Cooper, A. Treuille, J. Barbero, A. Leaver-Fay, K. Tuite, F. Khatib, A. C. Snyder, M. Beenen, D. Salesin, D. Baker, and Z. Popović. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 40–47, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-937-4. doi: 10.1145/1822348.1822354. URL `http://doi.acm.org/10.1145/1822348.1822354`.

[15] P. J. Denning. Is computer science science? *Commun. ACM*, 48(4): 27–31, Apr. 2005. ISSN 0001-0782. doi: 10.1145/1053291.1053309. URL `http://doi.acm.org/10.1145/1053291.1053309`.

[16] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, Apr. 2011. ISSN 0001-0782. doi: 10.1145/1924421.1924442. URL `http://doi.acm.org/10.1145/1924421.1924442`.

[17] J. Franklin. The formal sciences discover the philosophers' stone. *Studies in History and Philosophy of Science*, 25(4):513–533, 1994.

[18] W. Hodges. Logic and games. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2009 edition, 2009.

[19] P. G. Ipeirotis and P. K. Paritosh. Managing crowdsourced human computation: a tutorial. In *Proceedings of the 20th international conference companion on World wide web*, WWW '11, pages 287–288, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0637-9. doi: 10.1145/1963192.1963314. URL `http://doi.acm.org/10.1145/1963192.1963314`.

[20] A. Jaffe, A. Miller, E. Andersen, Y.-E. Liu, A. Karlin, and Z. Popovic. Evaluating competitive game balance with restricted play, 2012. URL `http://aaai.org/ocs/index.php/AIIDE/AIIDE12/paper/view/5470/5692`.

[21] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. Evaluating the crowd with confidence. Technical report, Stanford University, August 2012. URL `http://ilpubs.stanford.edu:8090/1051/`.

[22] D. Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011. ISBN 9781429969352. URL `http://books.google.com/books?id=ZuKTvERuPG8C`.

[23] L. Keiff. Dialogical logic. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2011 edition, 2011.

[24] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. ISBN 0321295358.

[25] J. Kulas and J. Hintikka. *The Game of Language: Studies in Game-Theoretical Semantics and Its Applications*. Synthese Language Library. Springer, 1983. ISBN 9789027716873. URL http://books.google.com/books?id=6GphI2_3u-sC.

[26] K. Lieberherr. The Scientific Community Game. Website, 2009. http://www.ccs.neu.edu/home/lieber/evergreen/specker/scg-home.html.

[27] K. J. Lieberherr and E. Specker. Complexity of Partial Satisfaction. *Journal of the ACM*, 28(2):411–421, 1981.

[28] K. J. Lieberherr and E. Specker. Complexity of Partial Satisfaction II. *Elemente der Mathematik*, 67(3):134–150, 2012. doi: 10.4171/EM/202. http://www.ccs.neu.edu/home/lieber/p-optimal/partial-sat-II/Partial-SAT2.pdf.

[29] K. J. Lieberherr, A. Abdelmeged, and B. Chadwick. The Specker Challenge Game for Education and Innovation in Constructive Domains. In *Keynote paper at Bionetics 2010, Cambridge, MA, and CCIS Technical Report NU-CCIS-2010-19*, December 2010. http://www.ccs.neu.edu/home/lieber/evergreen/specker/paper/bionetics-2010.pdf.

[30] M. Marion. Why Play Logical Games. Website, 2009. http://www.philomath.uqam.ca/doc/LogicalGames.pdf.

[31] J. Petit, O. Giménez, and S. Roura. Jutge.org: an educational programming judge. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, pages 445–450, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1098-7. doi: 10.1145/2157136.2157267. URL http://doi.acm.org/10.1145/2157136.2157267.

[32] A. Pietarinen. Games as formal tools vs. games as explanations. Technical report, 2000.

[33] D. Pink. *Drive: The Surprising Truth About What Motivates Us*. Canongate Books, 2011. ISBN 9781847677693. URL http://books.google.com/books?id=E0H_DIkg0I4C.

[34] K. R. Popper. *Conjectures and refutations: the growth of scientific knowledge, by Karl R. Popper*. Routledge, London, 1969. ISBN 710065078.

[35] W. Scacchi. The Future of Research in Computer Games and Virtual Worlds: Workshop Report. Technical Report UCI-ISR-12-8, 2012. http://www.isr.uci.edu/tech_reports/UCI-ISR-12-8.pdf.

[36] R. Sedgewick. The Role of the Scientific Method in Programming. Website, 2010. http://www.cs.princeton.edu/~rs/talks/ScienceCS.pdf.

[37] T. Tulenheimo. Independence friendly logic. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2009 edition, 2009.

[38] J. Väänänen. *Dependence Logic:*. London Mathematical Society Student Texts. Cambridge University Press, 2007. ISBN 9780521876599. URL http://books.google.com/books?id=KSR5xkAXiQAC.

[39] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM. ISBN 1-58113-702-8. doi: 10.1145/985692.985733. URL http://doi.acm.org/10.1145/985692.985733.

[40] K. J. S. Zollman. The communication structure of epistemic communities. *Philosophy of Science*, 74(5):574–587, 2007.

Appendix A Claim Language

A claim is an interpreted statement in first order predicate logic. A claim consists of an underlying model M, a predicate formula ϕ potentially containing free variables, an assignment g for the free variables in ϕ.

A `Formula` is either a simple `Predicate`, a `Compound` formula, a `Negated` formula, or a `Quantified` formula. A `Compound` formula consists of two subformulas, `left` and `right` and a `Connective` which is either an `And` or an `Or` connective. A `Quantified` formula consists of a `Quantification` and a subformula. A `Quantification` consists of a `Quantifier`, two `identifiers` representing the quantified variable name and type, and an optional `Predicate` further restricting the values the quantified variable can take. A `Quantifier` can be either a `ForAll`, an `Exists`, or `Free` which we use to declare free variables in a formula. Figure 1 shows the grammar for a formula expressed using the class dictionary notation [12].

```
Formula = Predicate | Compound | Negated |
    Quantified.
Predicate = <name> ident "(" <args> CommaList(
    ident) ")".
Compound = "(" <left> Formula
         <connective> Connective
         <right> Formula ")".
Negated = "(" "not" <formula> Formula ")".
Connective = And | Or.
And = "and".
Or = "or".

Quantified = <quantification> Quantification <
    formula> Formula.
Quantification = "(" <quantifier> Quantifier
               <var> ident
               "in" <type> ident
               <qPred> Option(
                    QuantificationPredicate) ")".
QuantificationPredicate = "where" <pred>
    Predicate.
Quantifier = ForAll | Exists | Free.
ForAll = "forall".
Exists = "exists".
Free = "free".
```

Figure 1. Formula Language

Appendix B Semantic Games

Given a claim `c` and two scholars, a verifier ver and a falsifier fal. Let M be the underlying model of `c`, let ϕ be the formula and g be `c`'s assignment to the free variables in ϕ. We define the semantic game of ver and fal centered around `c` `SG(c,`ver`,` fal`)` to be $G(\phi, M, g, ver, fal)$ which is a two-player, zero-sum game defined as follows:

1. If $\phi = R(t_1, ..., t_n)$ and $M, g \models R(t_1, ..., t_n)$, ver wins; otherwise fal wins.
2. If $\phi =!\psi$, the rest of the game is as in $G(\psi, M, g, fal, ver)$.
3. If $\phi = (\psi \wedge \chi)$, fal chooses $\theta \in \{\psi, \chi\}$ and the rest of the game is as in $G(\theta, M, g, ver, fal)$.
4. If $\phi = (\psi \vee \chi)$, ver chooses $\theta \in \{\psi, \chi\}$ and the rest of the game is as in $G(\theta, M, g, ver, fal)$.
5. If $\phi = (\forall x : p(x))\psi$, fal chooses an element a from M such that $p(a)$ holds, and the rest of the game is as in $G(\psi, M, g[x/a], ver, fal)$. If fal fails to do so, it loses.
6. If $\phi = (\exists x : p(x))\psi$, ver chooses an element a from M such that $p(a)$ holds, and the rest of the game is as in $G(\psi, M, g[x/a], ver, fal)$. If ver fails to do so, it loses.

The definition of G is adopted from the Game Theoretic Semantics (GTS) of Hintikka [25], [37]. We slightly modified Hintikka's

original definition to handle the quantification predicate in our language.