FSCP: A Platform for Crowdsourcing Formal Science

Ahmed Abdelmeged
Northeastern University
mohsen@ccs.neu.com

Karl Lieberherr
Northeastern University
lieber@ccs.neu.edu

ABSTRACT

We present the Formal Science Crowdsourcing Platform (FSCP). FSCP represents claims as interpreted predicate logic formulas. FSCP-based crowdsourcing systems focus a crowd of scholars on examining a family of claims to separate the true claims from false ones. Scholars examine a claim through participating in a substantiation game. Substantiation games are built on top of Hintikka's Game Theoretical Semantics (GTS). Furthermore, FSCP collects the defense and attack strategies on claims.

We also present an approach to evaluate scholars that we believe is new. We also present two approaches to estimate the truth likelihood of claims. We report on our experience with using an earlier version of FSCP in class.

Keywords

Crowdsourcing, Human computation, STEM innovation and education, epistemology, dialogic games, Karl Popper, mechanism design, social welfare, logic, defense strategies, games and quantifiers, virtual communities.

1. INTRODUCTION

Crowdsourcing contests have received a lot of attention in recent years. A crowdsourcing system is a generic system that enlists a crowd of users to help solve a problem defined by the system owners [11].

This paper presents the Formal Science Crowdsourcing Platform (FSCP), a highly configurable platform for constructing crowdsourcing systems for formal scientific knowledge. FSCP represents formal scientific knowledge as a set of claims. A claim is a predicate logic formula where all nonlogical symbols (i.e. constants, functions and predicates) are interpreted in a particular domain. Logical connectives are interpreted using the Game Theoretic Semantics (GTS) of Hintikka to yield two-person, zero-sum games, called refutation games (a.k.a semantical games). The two players are called the verifier and the falsifier. The existence of a winning strategy for the verifier means that the formula is true, the existence of a winning strategy for the falsifier means that the game is false.

To be submitted for review to the ACM Web Science 2013 Conference in Paris, May 2-4. 10 pages limit. Due Feb. 1, 2013

In FSCP, owners specialize the platform by creating labs. A lab is a crowdsourcing system that focuses a crowd of scholars on examining a family of claims to separate the true claims from false ones. Claim families are constructed by partially interpreting nonlogical symbols of a predicate logic formulas in a particular domain. Individual claims are obtained by completing the interpretation in the same domain. Scholars contribute to the lab by participating in refutation games that are syntactically derived from claims. By doing so, scholars provide evidence to the truth likelihood of individual claims in the lab. Furthermore, playing those games helps building the knowledge and intuition of individual scholars regarding the critical constructions of examples and counter examples embedded in the current winning strategies. The FSCP evaluates the performance of individual scholars as well as the truth likelihood of individual claims based on the history of all refutation games played in a particular lab.

1.1 FSCP Applications

FSCP has several **applications**, including:

1. problem solving and research in formal science. Funding agencies, such as NSF, define, in collaboration with interested researchers, labs that define the problem to be solved. Through playing the game, NSF builds a knowledge base of refutable claims and refutation attempts. Furthermore, the self-evaluating nature of FSCP will fairly evaluate the contributions of scholars and the collaborative nature will lead to productive team work. Newcomers can contribute by participating in a long-running lab (dozens of years).

2. teaching (traditional, online and massively open online) courses in STEM areas. To teach a particular problem solving skill, we design a lab for the problem. Playing FSCP challenge the students' self-image about their ability to solve the lab's problem. Thus, encouraging students to acquire the desired problem solving skill. The self-evaluating nature of FSCP helps lifting much of the evaluation from the teacher and allows stronger students to give precisely targeted feedback to weaker students.

3. software development for computational problems. A computational task is defined by a lab where the role of a scholar is played by an avatar (software). Competitions are held, and the winning avatars will contain the best (within this group of competing avatars) algorithms for the computational task.

1.2 Organization

This paper is organized as follows: In section 2 we present FSCP while in section 3 we present a novel approach for evaluating scholars and an approach for computing the truth likelihood of claims. In

section 5, we present our experience with SCG, a very close predecessor of FSCP. In section 6, we present some of the related work. Section 7 concludes the paper.

2. THE FSCP PLATFORM

A lab in FSCP consists of a claim family and a number of scholars. We begin by describing how claim families are specified in FSCP. Then we describe how refutation games and substantiation games are derived from claim families. Then we describe how scholars interact with the FSCP. Finally, we describe few approaches to derive scholar interactions in a lab.

2.1 Claim Families

A claim family consists of a logical formula and a model that provides an interpretation of all predicates mentioned in the formula. A claim consists of an assignment of values from the model to all free variables in the formula. Figure 1 shows the ClaimFamily and Claim structures.

```
final class ClaimFamily{
  final Formula f;
  final Model m;

  final class Claim{
    final Assignment g;
    ...
  }
}
```

Figure 1: ClaimFamily Structure

2.1.1 Formulas

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 2 shows the grammar for a formula expressed using the class dictionary notation [7].

2.1.2 Models

Models are used to interpret the types and predicates in a given formula. Figure 3 shows the Model interface. It has three methods. wellFormedTypeName checks whether a given type name is supported by the model. wellFormed checks whether a given value is a well formed value of a given type in the model. executePredicate executes a predicate provided by the model.

An example of a model is the SaddlePointModel shown in Figure 4. SaddlePointModel provides one type z1 which is a floating point number between 0 and 1 inclusive. wellFormedTypeName returns true only for "z1". wellFormed returns true for well formed values of type z1. The code for executePrediate supports a single predicate p(z1 x,z1 y,z1 q). This model can be used to interpret for the formula (free q in z1) (forall x in z1) (exists y in z1) p(x,y,q).

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

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

Figure 2: Formula Language

```
interface Model {
  boolean executePredicate(Assignment g,
      Predicate pred);
  boolean wellFormed(String value, String
      type);
  boolean wellFormedTypeName(String type);
}
```

Figure 3: Model Structure

```
class SaddlePointModel implements Model{
  public boolean wellFormedTypeName(String
      type) {
    return type.equals("z1");
  }
  public boolean wellFormed(String value,
      String type) {
    try{
      float v = Float.parseFloat(value);
      return v>=0 && v<=1;
    }catch(Exception e){
      return false;
    }
  }
  boolean executePredicate(Assignment g,
      Predicate pred) {
    if(pred.getName().getName().equals("p"))
        {

      float x = ...
      float y = ...
      float q = ...

      return (x*y + (1-x)*(1-y*y)) >= q;

    }else{
      throw new RuntimeException("Unknown␣
          predicate␣"+ pred.toString());
    }
  }
}
```

Figure 4: Sample Model

2.2 Scholars

The Scholar interface describes the inputs that FSCP collects from the crowd. The method decide is used to collect a decision from a scholar regarding whether (s)he wants to verify or falsify the given formula under the given model and assignment. Typically, a scholar would want to be the verifier of claims (s)he believes are true and be the falsifier of claims (s)he believes false. The method choose is used to collect an object from the given model for the quantification variable. Finally, the method propose is used to collect an assignment for free variables in the given formula other than the excluded assignments. It is possible to implement the Scholar interface such that it forwards requests to human scholars via email or a web interface for example. It is also possible to provide a self sufficient implementation that does not rely on human scholars. We call such Scholar implementations, avatars.

```
public interface Scholar {
  public enum Role {
    VERIFIER,
    FALSIFIER
  }
  String getName();
  Role decide(Formula f, Model m, Assignment
      g);
  String choose(Quantified f, Model m,
      Assignment g);
  Assignment propose(Formula f, Model m,
      Collection<Assignment> excluded);
}
```

Figure 5: Scholar Interface

2.3 Scholar Interaction

In FSCP, the interaction between scholars is centered around claims. Two scholars can interact by participating in a substantiation game. Substantiation games build on refutation games which we start explaining before we move to substantiation a refutation game or a substantiation game.

2.3.1 Refutation Games

Two scholars taking opposite positions on a specific claim c can participate in a refutation game denoted as c.RG(verifier, falsifier) where verifier is the scholar trying to support c and falsifier is the scholar disputing c.

Given a claim c and two scholars, a verifier ver and a falsifier fal. Let ϕ be the formula and M be the model of c's enclosing ClaimFamily. Let g be c's assignment to the free variables in ϕ. We define the refutation game of ver and fal centered around c c.RG(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 $G(\theta, M, g, ver, fal)$.

s1	s2	ref game	tested
v	v	c.RG(s1, s2)	s1
v	v	c.RG(s2, s1)	s2
f	f	c.RG(s1, s2)	s2
f	f	c.RG(s2, s1)	s1

Table 1: Scholar Under Test

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 [21], [34]. We slightly modified Hintikka's original definition to handle the quantification predicate in our language. The result of a refutation game consists of a record RGHistory of the two scholars verifier and falsifier, the winner, the assignment g, and a timestamp.

2.3.2 Substantiation Games

FSCP further extend the potential for interaction between scholars by allowing scholars to participate in test games even if they are taking the same positions on a specific claim c. Two scholars s1 and s2 taking two, not necessarily contradictory, positions r1 and r2 on claim c can participate in a substantiation game c.SG(s1, r1, s2, r2). If the two scholars hold contradictory positions on c, the substantiation game reduces to a refutation game. Otherwise, the substantiation game reduces to two refutation games c.RG(s1, s2) and c.RG(s2, s1) in which the two scholars teach each other. Given the two positions and the game, Table 1 can be used to identify the scholar being tested. It is important to identify the scholar under test for scholar evaluation purposes. The result of a substantiation game is a list of either one RGHistory record or two TestHistory records. A TestHistory extends RGHistory records with a underTest field.

2.4 Labs

An FSCP lab is a crowdsourcing system that consists of a ClaimFamily and a number of Scholars. Furthermore, based on its goal, a lab also provides:

1. system wide interaction mechanisms for scholars,

2. an evaluation mechanism for its scholars,

3. a mechanism for combining scholars' contributions.

We discuss system wide interaction mechanisms below and discuss scholar evaluation and combination of scholar contributions in Section 3.

It is possible to build several system wide interaction mechanisms on top of substantiation games and the Scholar interface. We give here three examples:

2.4.1 Battleship

Scholars independently propose claims to the system. When two scholars propose the same claim, the system collects their position and then engages them in a substantiation game.

2.4.2 Guided Search

The system chooses a claim and two scholars, then it collects their positions on that claim and engages them in a substantiation game. The system repeats until it reaches its goal. For example, suppose that we are building a crowdsourcing system to find the critical point of some free variable (from an ordered domain) in a formula. This is the value such that all claims above it are, for example, false. The system can effectively perform a binary search on the domain of that free variable. At each step in the binary search, the system creates a claim and chooses two scholars and engage them in a substantiation game.

2.4.3 Scientific Community Game

The Scientific Community Game (SCG) is a precursor to FSCP. The focus of SCG was educational. In SCG, scholars play a soccer-like tournament of binary matches. A match consists of an even number of rounds where scholars participate in binary games with alternating roles. SCG binary games are a precursor to substantiation games. In an SCG binary game, a scholar called the proponent proposes a claim. By doing so, the proponent is implicitly taking the verifier position on the claim it proposed. Then the other scholar, called the opponent, is asked to decide whether it agrees or disputes the claim. In either case, both scholars participated in a refutation game.

Eventually, a scholar ranking is produced as well as a trace of all refutation games. The intent was that scholars learn from the traces and the ranking is used to motivate them.

3. SCHOLAR EVALUATION

In a platform like FSCP, there are many qualities that we can measure and evaluate scholars based on. For example, we can measure the scholar's activity by counting the number of games it had played. We can measure the scholar's originality by counting the number of breakthroughs such as being the first to propose and verify a particular claim or being the first to falsify a claim. We can measure the scholar's learning during a particular period of time by comparing its rank at the beginning and at the end of that period.

Since the purpose of FSCP labs is to accurately classify claims. We are interested in measuring scholar's *reliability* which we define as scholar's likelihood of deciding to be the verifier of true claims and the falsifier of false claims. One approach to measure reliability is through testing the scholar's performance against a gold standard.

Instead, we measure scholar's *strength* which we define as the scholar's ability to push its opponent into self-contradiction. In Table 2, column "cont." provides an extensive list of the situations in which a scholar becomes self-contradictory. These situations can be summarized as losing a refutation game that is not played for testing as well as losing a refutation game while being the scholar under test. In both situations, the scholar contradicts its original position on the claim.

We prefer to measure *strength* instead of *reliability* because:

- strength is highly correlated to reliability. In order to increase their strength, scholars need to push their opponents into self-contradiction. To do so under the zero-sum nature of the game, scholars need to avoid falling intro self-contradiction themselves. To do so, scholars need to be reliable because unreliable scholars risk falling into self-contradictions. Especially, in the presence of other strong scholars. This can be seen from Table 2.

- It is free to accurately measure strength while it can be expensive to measure reliability. To directly measure reliability,

s1	s2	ref game	tested	winner	cont.	s1	s2
v	v	c.RG(s1, s2)	s1	s1	-	0	0
v	v	c.RG(s1, s2)	s1	s2	s1	0	1
v	v	c.RG(s2, s1)	s2	s1	s2	1	0
v	v	c.RG(s2, s1)	s2	s2	-	0	0
f	f	c.RG(s1, s2)	s2	s1	s2	1	0
f	f	c.RG(s1, s2)	s2	s2	-	0	0
f	f	c.RG(s2, s1)	s1	s1	-	0	0
f	f	c.RG(s2, s1)	s1	s2	s1	0	1
v	f	c.RG(s1, s2)	-	s1	s2	1	0
v	f	c.RG(s1, s2)	-	s2	s1	0	1
f	v	c.RG(s2, s1)	-	s1	s2	1	0
f	v	c.RG(s2, s1)	-	s2	s1	0	1

Table 2: Payoff Matrix

we need to either higher experts to provide a gold standard or have some claims redundantly examined by several scholars.

We designed the payoff matrix so that it only rewards a scholar with a point when the scholar successfully pushes its opponent into self-contradiction. This can be seen from Table 2.

4. MEASURING SCHOLAR STRENGTH

We present an algorithm to measure the strength of scholars that is independent of the order of the games based on the payoffs that scholars received. The function $Payoff(S_i, S_j)$ returns the number of times scholar S_i has pushed scholar S_j into self-contradiction.

$Str^{(-1)}(S_i) = 1$
$Wins^{(k)}(S_i) = \sum Payoff(S_i, S_j) * Str^{(k-1)}(S_j)$
$Losses^{(k)}(S_i) = \sum Payoff(S_j, S_i) * (1 - Str^{(k-1)}(S_j))$
$Str^{(k)}(S_i) = Wins^{(k)}(S_i)/(Wins^{(k)}(S_i) + Losses^{(k)}(S_i))$

The algorithm starts with an estimate of 1 for the strength of all scholars. Then it computes the weighted wins and losses for each player based on the payoffs and the strength of their opponents. 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.

The proposed algorithm has the following attractive properties:

- A scholar that never looses will have a strength of 1.
- A scholar that looses at least a single game, will have his/her strength hanging on the strength of other scholars.
- Loosing a game against a scholar with low strength will produce a large -ve impact, while loosing a game against a scholar with a high strength will have a small -ve impact.
- (The dual) Winning a game against a scholar with high strength will produce a large +ve impact, while winning a game against a scholar with low strength will have a small +ve impact.
- Order independence.

4.1 Truth Likelihood of Claims

The basic idea is to accumulate evidence about the truth and falseness of claims. Scholars provide evidence through picking positions on claims as well as through winning refutation games. Scholar strength is used to weigh their evidences.

For each claim c we associate two positive real numbers c_T and c_F with it. The higher c_T the more likely c is true. The higher c_F the more likely c is false. If $c_T > c_F$ then

$$L(c) = (c_T - c_F)/c_T$$

is the likelihood that c is true. And $1 - L(c)$ the likelihood that c is false. If $c_F \geq c_T$ then

$$L(c) = (c_F - c_T)/c_F$$

is the likelihood that c is false. and $1 - L(c)$ the likelihood that c is true.

At the beginning of a substantiation game for claim c, we adjust c_T and c_F as follows: For each scholar s taking the position of a verifier of c we add $str(s)$ to c_T. Similarly, for each scholar s taking the position of a falsifier of c we add $str(s)$ to c_F.

After a refutation game in which the verifier scholar ver loses, we add $str(ver)$ to c_F. Similarly, after a refutation game in which the falsifier scholar fal loses, we add $str(fal)$ to c_T. The intuition is that the losing player must have done its best to avoid the loss while the winning scholar might not have done its best to win.

5. 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.

5.1 Software Development

The most successful graduate classes were the ones that developed and maintained the software for SCG Court [1] 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 [19] where the best avatars needed to solve a constrained search problem using a modified Pascal triangle.

5.2 Algorithms

The most successful course (using [19] 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 exploratory and reformatory actions on piazza.com. We used a multi player 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 JASON 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-Shapely 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.

6. RELATED WORK

6.1 Crowd Sourcing and Human Computation

6.1.1 Dealing with Unreliable Workers

Most crowdsourcing systems must devise schemes to increase confidence in the worker's solutions to tasks, typically by assigning each task multiple times [16]. Larger et AL. present a general model for crowdsourcing tasks. In FSCP, because workers need to justify their answers in a game against another worker, unreliable workers will run into many contradictions and get a low rating. This means that their votes will minimally affect the final result, the knowledge base of true claims.

[8] is related to FSCP scholar ranking. The algorithm is an extended Bradley-Terry model called Crowd-BTU. The paper focuses on finding the quality of annotators in a crowdsourced setting. They study the exploration-exploitation tradeoff which is also relevant to FSCP for labeling claims as true or false.

The "Evaluating the Crowd with Confidence" paper [15] has a title that seems very applicable to FSCP. However, they use a model which is too simple for FSCP. In particular, in FSCP the errors depend on task difficulty, and worker errors are not independent of each other because they play a game.

6.1.2 Rating Systems

We use a rating system for games with wins, losses and draws. This subject has been studied for a long time and there are many applications of rating systems. For example, in chess and other sports, the Elo rating system is used. A good survey and critique of rating systems is given in [5]. Rating systems are a controversial subject and there are many algorithms that can be used. TopCoder [33] uses an Algorithm Competition Rating System to rank the coders.

6.1.3 Combining Worker's Contributions

In FSCP, we use two approaches to combine scholar contributions: (1) During the refutation games, the scholars give each other feedback by trying to drive each other into a contradiction. This is a collaboration which leads potentially to new ideas and knowledge fusion. (2) In FSCP, scholars vote on the truth or falseness of claims when deciding to verify or falsify claims. Furthermore, it is not enough for scholar to just vote but also it is important that they justify their votes through their actions in refutation games. We combine the votes with justifications into an overall vote for whether a claim is true. Related work is [8] and [16] which was already discussed above.

6.1.4 Competitions

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

The FSCP makes a specific, but incomplete proposal of a programming interface to work with the global brain [6]. What is currently missing is a payment mechanism for scholars and an algorithm to split workers into pairs based on their background.

The FSCP is a generic version of the "Beat the Machine" approach for improving the performance of machine learning systems [4].

Scientific discovery games, such as FoldIt and EteRNA, are variants of the FSCP. [9] describes the challenges behind developing scientific discovery games. [3] argues that complex games such as FoldIt benefit from tutorials. This also applies to the FSCP, but a big part of the tutorial is reusable across scientific disciplines.

6.1.5 Crowdsourcing complex tasks

[18] describes a general-purpose framework for solving complex problems through micro-task markets. Engaging in the scientific dialogs of FSCP could be done through a micro-task market. [28] proposes a language to define crowdsourcing systems. Our lab definition approach provides a declarative description of what needs to be crowdsourced.

[20] provides an interesting analysis of several issues relevant to FSCP: how incorrect responses should affect worker reputations and how higher reputation leads to better results.

6.2 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 [27], [13] have a long history going back to Socrates. The FSCP is an imperfect information game which builds on Paul Lorenzen's dialogical games [17].

6.3 Foundations of Digital Games

A functioning game should be deep, fair and interesting which requires careful and time-consuming balancing. [14] describes techniques used for balancing that complement the expensive playtesting. This research is relevant to FSCP lab design. For example, if there is an easy way to refute claims without doing the hard work, the lab is unbalanced.

6.4 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 FSCP has this property and provides a specific architecture for building knowledge bases in (formal) sciences. Collaboration between scholars is achieved through the scientific discourse implied by the refutation game. The information exchanged gives hints about how to play the game better next time. An interesting question is why this indirect communication approach works.

The NSF workshop report [31] discusses socio-technical innovation through future games and virtual worlds. The FSCP is mentioned as an approach to make the scientific method in the spirit of Karl Popper available to CGVW (Computer Games and Virtual Worlds).

6.5 Online Judges

An online judge is an online system to test programs in programming contests. A recent entry is [29] where private inputs are used to test the programs. Topcoder.com [33] includes an online judge capability, but where the inputs are provided by competitors. This dynamic benchmark capability is also expressible with the FSCP: The claims say that for a given program, all inputs create the correct output. A refutation is an input which creates the wrong result.

6.6 Educational Games

The FSCP 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 makes the scientific discourse more challenging. Another way is to pair the learner with another learner who is stronger. [2] 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.

6.7 Formal Sciences and Karl Popper

James Franklin points out in [12] 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 FSCP experiments are performed when the game associated with a claim is elaborated.

Karl Popper's work on falsification [30] is the father of non-deductive methods in science. The FSCP is a way of doing science on the web according to Karl Popper.

6.8 Scientific Method in CS

Peter Denning defines CS as the science of information processes and their interactions with the world [10]. The FSCP makes the scientific method easily accessible by expressing the hypotheses as claims. Robert Sedgewick in [32] stresses the importance of the scientific method in understanding program behavior. With the FSCP, we can define labs that explore the fastest practical algorithms for a specific algorithmic problem.

6.9 Games and Learning

Kevin Zollman studies the proper arrangement of communities of learners in his dissertation on network epistemology [35]. 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 FSCP is about learning unknown facts, namely new constructions.

6.10 SCG

SCG [24], [22], [23] is a close predecessor of FSCP. The original motivation for the SCG came from the two papers with Ernst Specker: [25] and the follow-on paper [26].

The key difference between FSCP and SCG is that SCG was targeted at evaluation of the scholars while FSCP is targeted at crowdsourcing true claims. FSCP is cleaner: there is a simple concept of self-contradiction and there is no longer a need to have the concept of strengthening a claim explicitly.

7. CONCLUSION AND FUTURE WORK

We presented FSCP, a crowdsourcing platform for formal science. FSCP provides a simple interface to a community that uses the (Popperian) Scientific Method.

We want to extend our model so that we can make claims about claims. For example, we want to have a "macro" for a claim to be optimal. We want to leverage claim relationships across labs and work with lab reductions as a useful problem solving tool.

We see a significant potential in putting the refutation-based scientific method into the cyberinfrastructure and make it widely available. We plan to, iteratively, improve our current implementation based on user feedback.

We see an interesting opportunity to mine the game histories and make suggestions to the scholars how to improve their skills to propose and defend claims. If this approach is successful, FSCP will make contributions to computer-assisted problem solving.

8. ACKNOWLEDGMENTS

We would like to thank Magy Seif El-Nasr, Casper Harteveld, Thomas Wahl and Tugba Koc for their input to the paper.

9. REFERENCES

[1] 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/ .

[2] 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.

[3] 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.

[4] 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.

[5] J. Beasley. *The Mathematics of Games*. Dover Books on Mathematics. Dover Publications, 2006.

[6] A. Bernstein, M. Klein, and T. W. Malone. Programming the global brain. *Commun. ACM*, 55(5):41–43, May 2012.

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

[8] X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *WSDM, Rome, Italy*, 2013.

[9] 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.

[10] P. J. Denning. Is computer science science? *Commun. ACM*, 48(4):27–31, Apr. 2005.

[11] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, Apr. 2011.

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

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

[14] A. Jaffe, A. Miller, E. Andersen, Y.-E. Liu, A. Karlin, and Z. Popovic. Evaluating competitive game balance with restricted play, 2012.

[15] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. Evaluating the crowd with confidence. Technical report, Stanford University.

[16] D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *NIPS*, pages 1953–1961, 2011.

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

[18] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut. Crowdforge: crowdsourcing complex work. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 43–52, New York, NY, USA, 2011. ACM.

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

[20] M. Kosinski, Y. Bachrach, G. Kasneci, J. V. Gael, and T. Graepel. Crowd iq: measuring the intelligence of crowdsourcing platforms. In *WebSci'12*, pages 151–160, 2012.

[21] J. Kulas and J. Hintikka. *The Game of Language: Studies in Game-Theoretical Semantics and Its Applications*. Synthese Language Library. Springer, 1983.

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

[23] K. J. Lieberherr and A. Abdelmeged. 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.

[24] 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 .

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

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

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

[28] P. Minder and A. Bernstein. Crowdlang - first steps towards programmable human computers for general computation. In *Proceedings of the 3rd Human Computation Workshop*, AAAI Workshops, pages 103–108. AAAI Press, 2011.

[29] 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.

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

[31] 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.

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

[33] TopCoder. The TopCoder Community. Website, 2009. http://www.topcoder.com/.

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

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