

Collusion-Resistant Evaluation of Side-Choosing Games

Ahmed Abdelmeged

Karl Lieberherr

Ruiyang Xu

*College of Computer and Information Science,
Northeastern University,
440 Huntington Avenue,
202 West Village H,
Boston, MA 02115, USA*

MOHSEN@CCS.NEU.EDU

LIEBER@CCS.NEU.EDU

RUIYANG@CCS.NEU.EDU

9-22-2014

Abstract

Very OLD.

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. Introduction to Side-Choosing Games

(?) Side-choosing games are a new kind of game where the first move is simultaneous and involves choosing a side on a position p that comes from another two-person game without draws, called a reference game. The choice is about two possibilities:

- p is a winning position, i.e., there is a winning strategy, and,
- p is a losing position, i.e., there is no winning strategy.

We call the first decision to take the *proponent* role and the second decision to take the *opponent* role on p .

Side-choosing games model debates on a topic p . Either you are for p (proponent) or you are against p (opponent). Your choice must be an informed choice: you must be ready to play an actual game from position p and perform according to your side choice. When the reference game p is played between two participants, one participant is in proponent role and the other in opponent role. If you are proponent you must win in the proponent role and if you are opponent you must win in the opponent role. If participant x does not perform according to its choice, we say that the participant was brought into a contradiction. We use the impersonal reference "it" because participant may just be software.

We explicitly model the situation where both participants want to be proponent or both want to be opponent on position p . Because the game requires one participant in proponent role and the other in opponent role, we put one participant in devil’s advocate role by forcing him or her to take the opposite side that was chosen. Therefore, we record game results using the following kind of table:

Make a Latex table

	W	L	Forced
1	x	y	x
2	x	y	y
3	x	y	none
4	y	x	x
5	y	x	y
6	y	x	none

Column W is for winners, column L for losers and column $Forced$ indicates who was forced (played in devil’s advocate role). The winner or loser may be forced or *none*. In debate 1, x won in forced side while y lost in non-forced side. This is not a good outcome for y and we say y had a fault because it lost in non-forced side. Also in debate 3, participant y had a fault.

Why are side-choosing games interesting? They model many interesting practical debate situations. For example, the position p could be a claim formulated in a specific structure using a logical sentence. Side-choosing games are very simple: they are based on the concepts of winning, losing and forced, yet we can prove useful theorems about them. Those theorems allow us to make informed choices in many different application domains.

What is the nature of the theorems that we prove. They are theorems about tables (like the table above) and their mappings to rankings. A ranking is a total relation between the participants. The theorems make predictions about adding rows with specific properties to the tables and how they affect the rankings. These predictions are useful because they allow us to predict that certain rating changes are impossible.

1.1 Participant Evaluation

Our objective is to find the most meritorious participants given a table of debate results. An alternative would be to go for the positions that are predominantly successfully defended or opposed. But we prefer the meritocratic approach which is more robust for reasons that we don’t discuss in this paper. Once we have the most meritorious participant we can use it to decide and defend/refute the claims.

The first approach is a centralized approach where there is a centralized participant that “knows the truth” and that will evaluate the regular participants. The centralized approach has the problem that it is costly to develop the centralized participant.

We prefer a distributed approach which uses games between the participants. However there are a few obstacles to overcome. When the participants evaluate each other there are a few dangers: (1) the evaluation may not be objective. The participants might have favorites. (2) multiple participants might collude to damage another participant. For (2) we have collusion-resistant ranking functions and for (1) we use contradiction-agreement games. But our collusion-resistant ranking functions

create their own problem for which we have a solution too: The collusion-resistant ranking functions make the players hide in a corner avoiding to play debates. The collusion-resistant ranking functions don't incentivize the participants to play. We use a scheduler to force the debates that are needed for a fair evaluation.

1.2 Prominent Application

Semantic games

2. Related Work

2.1 Side-Choosing Games

Many different game forms have been studied but the general concept of a side-choosing game seems to be new.

http://www.phil.uni-mannheim.de/fakul/phil2/rueckert/pdf/WDL_Rohfassung.pdf

In semantic games, the notion of choice and supporting the choice one has made is an important theme.

Dialogical Logic. Erlangen School founded by Paul Lorenzen. Methodical Constructivism.

But the abstraction of a side-choosing game was not needed back then because there was no web that makes it easy to execute, record and check such games. Back then there was no need for tournaments of such games and there was no danger of collusion.

Our concepts of forcing, control and collusion-resistance are new and so is our axiomatic theory involving those concepts.

2.2 Axiomatic Treatment of Ranking Functions

In (Rubinstein, 1980), Rubinstein provides an axiomatic treatment of tournament ranking functions that bears some resemblance to ours. Rubinstein's treatment was developed in a primitive framework where "beating functions" are restricted to complete, asymmetric relations. Rubinstein showed that the points system, in which only the winner is rewarded with a single point is *completely* characterized by the following three *natural* axioms:

- anonymity which means that the ranks are independent of the names of participants,
- positive responsiveness to the winning relation which means that changing the results of a participant p from a loss to a win, guarantees that p would have a better rank than all other participants that used to have the same rank as p , and
- Independence of Irrelevant Matches (IIM) which means that the relative ranking of two participants is independent of those matches in which neither is involved.

Our LCE axioms are, in some sense, at least as strong as Rubinstein's IIM because, according to LCE, the relative rank of some participant p_x w.r.t. another participant p_y cannot be worsened by games that p_x does not participate in nor can it be improved by games that p_y does not participate in.

In (Gonzalez-Daz, Hendrickx, & Lohmann, 2014), the authors provide an axiomatic study of several ranking functions that are based on rating methods. Eight ranking methods, including the points system, and fourteen different axioms, including Rubinstein’s IIM, are studied in the paper. Each of the ranking methods is analyzed to determine the axioms it possesses. Only the points system possesses the IIM axiom. The IIM axiom is, however, considered to be an undesirable axiom to have because it is thought to prevent the ranking function from making up for any advantage given to certain participants by a tournament schedule that contains only a subset of the games that would be played in a round robin tournament. Again, none of these ranking functions is specifically developed for SG tournaments. Also, we use a round-robin-like tournament and IIM-like axioms are not undesirable to have.

2.3 Tournament Ranking Functions

Rating methods can be used to rank tournament participants. There is a vast body of literature on the topic of *heuristic* (Beasley, 2006) rating methods aiming to estimate the skill level of participants such as the Elo (Elo, 1978) rating method. (Langville & Meyer, 2012) gives a recent comprehensive overview of rating methods used in sports tournaments. Our work differs from this vast body of literature in two important aspects. First, our axioms and ranking method are the first to be developed for an extended framework that we developed specifically to capture some of the peculiarities of SG tournaments such as side choice and forcing. Second, our work is the *first* to be concerned with collusion resilience.

3. Formal Definitions

We are defining the terminology for a new field of side-choosing games. We better choose the terminology carefully so that it will be adopted by others. Proposed renamings from the dissertation:

OLD NEW

LCE CR (collusion-resistant)

beating function SCG-table

NNRW NNEW

NPRL NPEL

Regard Effect

plan for ordering definitions

SCG and SCG-tables

Total Preorders

Scoring Functions

Rankings

Axioms

NNEW

NPEL

CR

Monotonicity Constraints

NNEW and NPEL using monotonicity constraints

Defining CR using monotonicity constraints
Equivalence to previous CR definition

The paper is organized as follows: We first describe the challenges behind the problem that we solve and we give a heuristic argument for the result we prove formally. In section ?? we give the key definitions from SCG, SCG-tables, axioms and monotonicity constraints and how we use them to express the axioms. Section ?? describes the main result, a representation theorem for ranking functions satisfying the axioms. We give two proofs, one based on monotonicity constraints and the other on the analysis of a Venn diagram describing game results. In section ?? we apply side-choosing games to organize competitions using semantic games, a special kind side-choosing game. (Let's see whether we have enough room to describe this for the AAAI conference)

3.1 Side-Choosing Games

Side-ChoosingGame = Side-Choice x GeneralSemanticGame
GeneralSemanticGame: details of protocol not important
Game outcome must satisfy certain rules
Winning strategies

Examples of families of semantic games
Logics
concrete example: Integer inequality.
Positions in explicit-form games
concrete example: Chess: mate in two.

Meritocracy management
Tables with base and derived fields
Important table: SCG-Table: (W, L, F)
SCG-Rows
How to get to the SCG-Table?

Side-choosing games come from a generalization of semantic games with side-choice mechanism. Before we give out the definition of side-choosing games, we shall first have a look at a definition of semantic games in Logic and Epistemology (by A-V Pietarinen):

DO WE STILL NEED THIS DEFINITION?

- Two players confront one another, facing a claim and observing a set of rules telling them which moves are legal.
- They both try to win the game by proving any side of claim (i.e. be a proponent or opponent of the claim), and if one of them finds a systematic way of doing so, he or she has a winning strategy.
- The set of game rules is fixed by the logically active components in language. In the case of first-order languages the logically active components comprise the existential and universal quantifiers.

Generally speaking, A game is defined by game states, moves and the rules when the moves are applicable. The definition of side-choosing games is a generic definition built on mapping logical sentences in predicate-logic to games so that a property holds about truth of the sentences and the existence of winning strategies. The outcome of a side-choosing game gives the winner, looser and information about who was forced.

Side-choosing game consists of two parts:

- Simultaneous design-time decision, this is the design time for the winning strategy for Proponent/Opponent.
- Run-time decision (imposed from outside) followed by sequential game defending run-time decision. This is the decision used when the defense game executes and might involve forcing at most one participant. Specifically, once both players take same side, then choose one of the two players randomly. Here we use a state variable: `already_forced`. Goal is to force one by one and play two games. So if there are two "not `already_forced`", choose one of the two players, and force the player; if one "not `already_forced`", force that one.

Next, we discuss the side-choice mechanism in detail here: since we don't want both players choosing the same side, that means, given a claim, one of them must be a proponent and the other be a opponent, one of them must be forced to change to the other side while both of them have found the winning strategy for the same side. Briefly speaking, if both the players have chosen the same side, a Run-time decision must be imposed.

Here we introduce the Agreement Map: The agreement function maps two players x, y with their design-time side choices (d_x, d_y) into a set of games between the two players with run-time side choices (r_x, r_y) . A specific agreement function is called CAG (Competitive Agreement Game), which is designed for tie-break when both players have chosen the same side ($d_x = d_y$); We choose the forced player randomly to balance the potential disadvantage. Then we play two games to give each player a chance to test the other:

- Choose a design-time side: P or O: d_x, d_y for G . P means: The start position of G is a winning position for player in P role. O means: The start position is not a winning position for player in P role.
- If $d_x \neq d_y$: play game $SCG(G, x, y, d_x, d_y, d_x, d_y)$
- Else: Use agreement map AM to determine games to be played. Each game is of the form $SCG(G, x, y, d_x, d_y, r_x, r_y)$ for run-time side-choices r_x, r_y determined by the agreement map AM.

We record the game results with only the information of who was the proponent and who won the game. There are actually four cases of game result in terms of any specified play x (Figure 1):

P (Proponent)	Winner
x	x
x	!x
!x	x
!x	!x

Figure 1: Semantic game result cases in terms of player x

Considering the side-choice mechanism, the original side chose by each player should also be recorded. Then combined with game result information above, one can infer who has been forced. It's easy too see that there are only four cases of side-choice, and if both players chosen the same side, one of them must be forced to change side (Figure 2):

Sx (Side(x))	Slx (Side(lx))
P	O
O	P
P	P
O	O

Figure 2: Side-choice cases in terms of player x and not x

Given the game result and side-choice information, one can derive further information about the game such as who is the loser? Is there anyone forced? Who is forced? We call this game result table with derived information as Concrete SCG-Table. And since we only care about the information of winner, loser and forced side-choice, dropping out other information we abstract the Concrete SCG-Table to what we want, and we called the abstracted new table SCG-table (Figure 3):

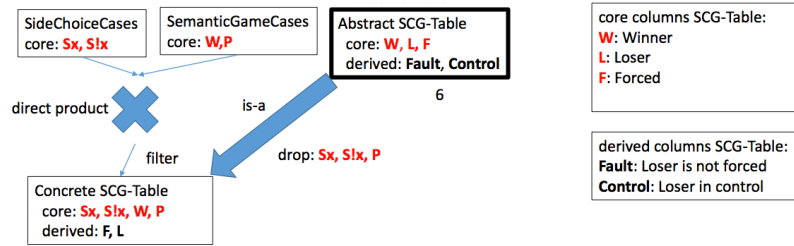


Figure 3: How SCG-table derived from semantic games and side-choice

Now we can give out the definition of side-choosing game:

- SCG is a tuple: (CG,AM), where CG is a two-person combinatorial game and AM is an Agreement Map.
- The game is between two players and starts with a position (node) q in CG for which the two players choose a side.
- Either a single game is played (different sides) or AM is activated (same sides) and one or more games are played.
- The results of the games are recorded in the form W,L,F.

Each game result then produced as an SCG-row in an SCG-table. It's reasonable to show that the game out comes shall follow certain rules (SCG-Table Rules), if it's a side-choosing game:

- Columns are: W,L,F (for Winner, Loser, Forced) (SCG-Table Rule 0).

- W,L contain either one of the player but $W=L$. There are no ties (SCG-Table Rule 1).
- F contains none or W or L. A participant is forced if it has to take the opposite side (SCG-Table Rule 2).
- Multiple rows may involve the same two participants (SCG-Table Rule 3)
- Each row presents only one game result between two distinct participants in players set (SCG-Table Rule 4).

One thing to notice is that the SCG-table definition does not specify any legal movement (i.e. game protocols is a separation of concerns) for the players. Hence the game protocol language must guarantee that whenever a claim is true, the expression of winning strategy via the protocol language is possible, in other words, there exists a mapping SCG from claims to side-choosing games so that the side-choosing game for a given claim $\langle \Phi, A \rangle$ is a game played by a run-time proponent x and a run-time opponent y , denoted $SCG(\langle \Phi, A \rangle, x, y)$, such that there exists a winning strategy for participant x defending claim, given structure A (denoted as $A \models \Phi$). Obviously, for standard semantic games, the protocol language always guarantees this. see Figure 4).

Φ	Move	Subgame
$\forall x : \Psi(x)$	<i>fal</i> provides x_0	$SG(\langle \Psi[x_0/x], A \rangle, ver, fal)$
$\Psi \wedge \chi$	<i>fal</i> chooses $\theta \in \{\Psi, \chi\}$	$SG(\langle \theta, A \rangle, ver, fal)$
$\exists x : \Psi(x)$	<i>ver</i> provides x_0	$SG(\langle \Psi[x_0/x], A \rangle, ver, fal)$
$\Psi \vee \chi$	<i>ver</i> chooses $\theta \in \{\Psi, \chi\}$	$SG(\langle \theta, A \rangle, ver, fal)$
$\neg \Psi$	N/A	$SG(\langle \Psi, A \rangle, fal, ver)$
$p(x_0)$	N/A	N/A

Figure 4: Standard semantic game, where ver = Proponent and fal = Opponent

Next, for illustrating purpose, we offer two simple examples of side-choosing games. In the first one we use Chess as reference game and in the second we use the semantic game behind a logical formula as the reference game.

- **Chess game: Mate in 2**

We start with a chess position similar to one chosen from Figure 5. And one player, the Proponent, claims to mate in 2 and the other, the Opponent, claims to prevent a mate in two. To determine the game outcome, the players play from the given position to support their claim.



Figure 5: a checkmate puzzle from <http://www.chess.com>

- **Semantic game: Saddle Point**

In the second example, we use the logical formula:

$$\forall x \in [0, 1] \exists y \in [0, 1] : x \cdot y + (1 - x) \cdot (1 - y^2) \geq c$$

for some $c \in [0, 1]$, e.g., $c = 0.615$. Behind every formula in predicate logic there is a semantic game between two players, one playing the Proponent role and the other the Opponent role. The Opponent chooses the \forall quantifier values and the Proponent the \exists quantifier values. The Proponent wins if the atomic formula to the right of $:$ is true after the quantified variables have been assigned. The above sentence for $c = 0.615$ is true and therefore the Proponent has a winning strategy, for example by using the Skolem function $y = x$.

3.2 Algorithms to Analyze

\leq_{WC} is **defined** as: $x \leq_{WC} y$ if $wins(x) \geq wins(y)$.

\leq_{FC} is **defined** as: $x \leq_{FC} y$ if $faults(x) \leq faults(y)$.

We want to know whether the algorithms \leq_{WC} and \leq_{FC} are collusion-resilient.

3.3 Property to Check

Definition of collusion resilience: A ranking \leq of participants is said to be collusion-resilient if for all debate tables D and for any two participants x and y , the property $x \leq^D y$ implies $x \leq^E y$, where E is D with added debates g that x cannot control, i.e., for which $!g.control(x)$.

Informally, if $x \leq^D y$ and more debates that x cannot control are added to D resulting in E , then $x \leq^E y$ holds.

3.4 Choose your side and defend it

3.4.1 QUESTION 1

Is \leq_{WC} collusion resilient?

3.4.2 QUESTION 2

Is \leq_{FC} collusion resilient?

Justify your answer by providing a proof why the algorithm has the property or not.

Hint: If you look for a positive answer, the proof is rather short. If you look for a negative answer, it is enough to consider a small number of participants only.

3.5 Ranking Functions

4. The Challenge of Confusion

A tournament can exhibit confusing behavior which makes it difficult to extract the most meritorious player. The following confusing behaviors all may be present in a tournament. In general, we don't know whether a claim is true or false.

- the correct side may be chosen, yet the defense will fail or, the incorrect side may be chosen, yet the defense will succeed. In other words: true claims may be refuted and false claims may be defended.
- players may lie about their strength and lose on purpose to help their colluding partner.

The challenge is how to find order among all those confusing events.

An issue related to confusion is the garbage-in garbage-out property of SCG tournaments. If all players are weak, we cannot expect a useful outcome.

5. Informal Reasoning about Ranking Mechanisms

Before we give a formal proof why it is best to minimize unforced losses to securely rank players, we give a heuristic argument. Consider Table 2 which defines 4 counting functions for SCG-tables: For a player p : $WF(p)$, $WU(p)$, $LF(p)$ and $LU(p)$. Let's analyze those 4 basic functions:

- **WF.** Winning in forced side is an achievement but the loser could have lied about his/her strength and lost intentionally to benefit the winner. Therefore, WF could be abused and maximizing WF is not a reliable indicator of strength.
- **WU.** Winning in unforced side is an achievement but the same argument as for WF applies. In addition, the loser could be forced in which case winning in unforced side is not a sign of strength. Therefore maximizing WU is not a reliable indicator of strength.
- **LF.** When you lose while forced you cannot be blamed. If your side choice was correct, it is expected that you lose unless the opponent is weak. Therefore, LF is not an indicator of weakness and minimizing LF is not a reliable indicator of strength.
- **LU (Fault).** Losing in unforced side is clearly a mistake. The loser got into a contradiction independent of whether the loser is proponent or opponent. Therefore, minimizing LU is a reliable indicator of strength.

The above argument is imprecise but gives a first hint why counting unforced losses is the best approach.

We now turn to two formal approaches to prove that fault counting plays a central role. The first approach uses reasoning about monotonicity constraints of functions and the second approach from (?) is using case analysis of a Venn diagram showing all possible game outcomes for two players.

6. Reasoning about ranking mechanisms via a general form scoring function

Inspired by the utility theory from Artificial Intelligent, we assigned each palyer a score, which comes from a uniformed utility function, or scoring function. Notice that all the infomation comes form the raw competition result table, and they can be classified into different cases (as shown in Figure 6). We find the less general proof using structured ranking based on scoring easier and more revealing.

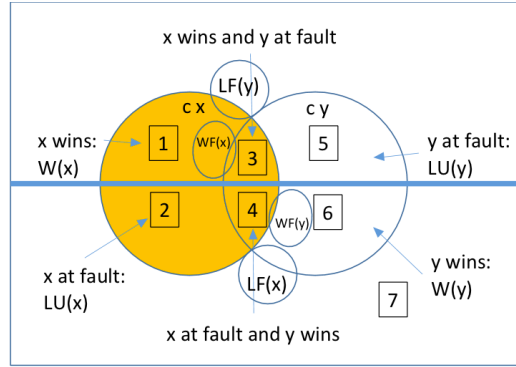


Figure 6: Venn diagram for all possible game classification in terms of player x and y.

Winner	Loser	Forced
p_x	p_y	None
p_x	p_y	p_y
p_y	p_x	p_y
p_x	p_z	p_x

Table 1: snippet of debate results table.

Case	Win/Lose	Forced/Unforced
WF	Win	Forced
WU	Win	Unforced
LF	Lose	Forced
LU(Fault)	Lose	Unforced
NP	Not participated	Not participated

Table 2: It can be shown that for every debator, there are only 5 cases in each debate.

Therefore, given a results table T and a specified individual(p_x), there are actually 5 kinds of basic statistic information could be retrieved, i.e.

$$WF(p_x), WU(p_x), LF(p_x), LU(p_x) \text{ and } NP(p_x)$$

Based on those information, one can define a ranking function:

$$U_{p_x} = U(WF(p_x), WU(p_x), LF(p_x), LU(p_x), NP(p_x))$$

Notice that, given the total number $|T|$ of games in the result table, we have:

$$NP(p_x) = |T| - WF(p_x) - WU(p_x) - LF(p_x) - LU(p_x)$$

Therefore U_{p_x} can be rewritten as:

$$U_{p_x}^T = U(WF(p_x), WU(p_x), LF(p_x), LU(p_x), |T|)$$

And we should also specify that: p_x is weakly better than p_y under table T iff $U_{p_x}^T \leq U_{p_y}^T$. Then we could reform the *NNEW*, *NPEL* and *CR* ranking axioms in terms of scoring function as following:

NNEW. For any two players p_x and p_y , given a table T , if T' comes from one more game, where p_x 's won, added into T , then we have: $U_{p_x}^T \leq U_{p_y}^T \Rightarrow U_{p_x}^{T'} \leq U_{p_y}^{T'}$

NPEL. For any two players p_x and p_y , given a table T , if T' comes from one more game, where p_y 's lost, added into T , then we have: $U_{p_x}^T \leq U_{p_y}^T \Rightarrow U_{p_x}^{T'} \leq U_{p_y}^{T'}$

CR. For any two players p_x and p_y , given a table T , if T' comes from one more game, where p_x 's out of control (which means either p_x has not participated or p_x was forced to lose), added into T , then we have: $U_{p_x}^T \leq U_{p_y}^T \Rightarrow U_{p_x}^{T'} \leq U_{p_y}^{T'}$

For convenience of analysis, certain monotonicity notations should be introduced (Notice that we're just borrowing differential notations here, but, in fact, since function U is discrete, it cannot be differentiated):

$$\partial_X U \geq 0 : U \text{ is monotonically non-decreasing on } X$$

$$\partial_X U \leq 0 : U \text{ is monotonically non-increasing on } X$$

$$\partial_X U = 0 : U \text{ is indifference on } X$$

$$\text{Where } X \in \{WF, WU, LF, FU, |T|\}$$

Being provided with the definitions and notations above, one can start reason about the inner monotonical properties of the scoring function. Suppose we have a result table T , and from the table we have two players p_x and p_y , where $U(p_x) \leq U(p_y)$. And this inequality shall hold under *NNEW*, *NPEL* and *CR*, which are the ranking axioms.

First, we can prove that the information of $|T|$ could be eliminated from the scoring function, if it's *CR*; Based on *CR* axiom, we consider a special case in which we add more games to a given result table T_0 where neither p_x nor p_y have participated. That means, in this special case, one can regard the scoring function of p_x and p_y as a function of only one variable $|T|$:

$$U_{p_x}^T = U_{p_x}^T(|T|)$$

$$U_{p_y}^T = U_{p_y}^T(|T|)$$

then we define a new comparison function:

$$R_{xy}(|T|) = U_{p_x}(|T|) - U_{p_y}(|T|)$$

now suppose we have $U_{p_x}(|T_0|) \leq U_{p_y}(|T_0|)$. Since CR needs this inequality maintained unchanged during more games being added, in which both p_x and p_y are out of control, we have:

$$\partial_{|T|} R_{xy}(|T|) \leq 0 \text{ for } |T| \geq |T_0|$$

however one can also suppose that $U_{p_x}(|T_0|) \geq U_{p_y}(|T_0|)$. And based on CR for this case, we have the contrary constraint:

$$\partial_{|T|} R_{xy}(|T|) \geq 0 \text{ for } |T| \geq |T_0|$$

then one can obtain

$$\partial_{|T|} R_{xy}(|T|) = \partial_{|T|} (U_{p_x}^T - U_{p_y}^T) = 0 \quad (1)$$

that means the effect of increasing the size of table is independent with the comparison of score between any two players. This property gives us a good reason to ignore the $|T|$ (as well as NP) in future analysis, which allows us to redefine the form of scoring function as:

$$U_{p_x}^T = (WF(p_x), WU(p_x), LF(p_x), LU(p_x))$$

Next, we shall see a scoring function is $NNEW$, if and only if we add one more game, where p_x won and p_z lose, to the result table T to form a new table T' , and the inequality $U_{p_x}^{T'} \leq U_{p_y}^{T'}$ unchanged, which means we have either:

$$\begin{aligned} & U_{p_x}^T(WF(p_x) + 1, WU(p_x), LF(p_x), LU(p_x)) \\ & \leq U_{p_y}^T(WF(p_y), WU(p_y), LF(p_y), LU(p_y)) \end{aligned}$$

or:

$$\begin{aligned} & U_{p_x}^T(WF(p_x), WU(p_x) + 1, LF(p_x), LU(p_x)) \\ & \leq U_{p_y}^T(WF(p_y), WU(p_y), LF(p_y), LU(p_y)) \end{aligned}$$

Hence we reform $NNEW$ in terms of the monotonicity of U :

$$\begin{aligned} \partial_{WF} U & \leq 0 \\ \partial_{WU} U & \leq 0 \end{aligned} \quad (2)$$

Similarly, a scoring function is $NP EL$, if and only if we add one more game, where p_y lose and p_z won, to the result table T to form a new table T' , and the inequality $U_{p_x}^{T'} \leq U_{p_y}^{T'}$ unchanged, which means we have either:

$$\begin{aligned} & U_{p_x}^T(WF(p_x), WU(p_x), LF(p_x), LU(p_x)) \\ & \leq U_{p_y}^T(WF(p_y), WU(p_y), LF(p_y) + 1, LU(p_y)) \end{aligned}$$

or:

$$\begin{aligned} & U_{p_x}^T(WF(p_x), WU(p_x), LF(p_x), LU(p_x)) \\ & \leq U_{p_y}^T(WF(p_y), WU(p_y), LF(p_y), LU(p_y) + 1) \end{aligned}$$

Hence we reform $NPEL$ in terms of the monotonicity of U :

$$\begin{aligned}\partial_{LF}U &\geq 0 \\ \partial_{LU}U &\geq 0\end{aligned}\tag{3}$$

Finally, we come again to analyze the axioms of CR . We shall see that a scoring function is CR if and only if the following two cases hold.

The first case: we add one more game, where p_x not participated, to the result table T to form a new table T' , and to maintain the inequality $U_{p_x}^{T'} \leq U_{p_y}^{T'}$ unchanged, we have 5 cases:

$$\begin{aligned}U_{p_x}^T(WF(p_x), WU(p_x), LF(p_x), LU(p_x)) \\ \leq U_{p_y}^T(WF(p_y) + 1, WU(p_y), LF(p_y), LU(p_y))\end{aligned}$$

or:

$$\begin{aligned}U_{p_x}^T(WF(p_x), WU(p_x), LF(p_x), LU(p_x)) \\ \leq U_{p_y}^T(WF(p_y), WU(p_y) + 1, LF(p_y), LU(p_y))\end{aligned}$$

or:

$$\begin{aligned}U_{p_x}^T(WF(p_x), WU(p_x), LF(p_x), LU(p_x)) \\ \leq U_{p_y}^T(WF(p_y), WU(p_y), LF(p_y) + 1, LU(p_y))\end{aligned}$$

or:

$$\begin{aligned}U_{p_x}^T(WF(p_x), WU(p_x), LF(p_x), LU(p_x)) \\ \leq U_{p_y}^T(WF(p_y), WU(p_y), LF(p_y), LU(p_y) + 1)\end{aligned}$$

or y also didn't participate in the added game, that means nothing will change. This gives us monotonicity constrains:

$$\begin{aligned}\partial_{WF}U &\geq 0, \partial_{WU}U \geq 0, \\ \partial_{LF}U &\geq 0, \partial_{LU}U \geq 0\end{aligned}\tag{4}$$

The second case: we consider the case where p_x was forced to lose the game. We add one more game, where p_x forced to lose and either p_y or p_z won the game, to the result table T to form a new table T' , and to maintain the inequality $U_{p_x}^{T'} \leq U_{p_y}^{T'}$ unchanged, we must have either:

$$\begin{aligned}U_{p_x}^T(WF(p_x), WU(p_x), LF(p_x) + 1, LU(p_x)) \\ \leq U_{p_y}^T(WF(p_y), WU(p_y) + 1, LF(p_y), LU(p_y) + 1)\end{aligned}$$

or:

$$\begin{aligned}U_{p_x}^T(WF(p_x), WU(p_x), LF(p_x) + 1, LU(p_x)) \\ \leq U_{p_y}^T(WF(p_y), WU(p_y), LF(p_y), LU(p_y) + 1)\end{aligned}$$

This gives us the monotonicity constrain:

$$\partial_{LF}U \leq 0\tag{5}$$

Combining constrains of 5 with 4, we have reformed CR in terms of the monotonicity of U :

$$\begin{aligned}\partial_{WF}U &\geq 0, \partial_{WU}U \geq 0 \\ \partial_{LF}U &\geq 0, \partial_{LU}U = 0\end{aligned}\tag{6}$$

In summation, the analysis above is enough to determine the monotonicity of scoring function from ranking axioms, and we can reform the axioms with the monotocity of the scoring function as following:

$$\begin{aligned}
NNEW : \partial_{WF}U &\leq 0, \partial_{WU}U \leq 0 \\
NPEL : \partial_{LF}U &\geq 0, \partial_{LU}U \geq 0 \\
CR : \partial_{WF}U &\geq 0, \partial_{WU}U \geq 0, \partial_{LF}U = 0, \\
&\partial_{LU}U \geq 0, \partial_{|T|}(U_{p_x}^T - U_{p_y}^T) = 0
\end{aligned}$$

And if we define a new property of Local Fault Based (LFB), which simply means that the comparison of scores between any two players only depend their fault counting, as the combination of constrains in 1, 6 and 2:

LFB.

$$\begin{aligned}
\partial_{WF}U &= 0, \partial_{WU}U = 0, \\
\partial_{LF}U &= 0, \partial_{|T|}(U_{p_x}^T - U_{p_y}^T) = 0
\end{aligned}$$

Then we can easily derive the following relations among those axioms and the *LFB* property:

$$\begin{aligned}
NNEW \wedge CR &\Rightarrow \text{LFB} \\
NPEL \wedge \text{LFB} &\Rightarrow CR \\
NNEW \wedge NPEL &\Rightarrow (CR \Leftrightarrow \text{LFB})
\end{aligned}$$

In the next section, we shall see that these relations could be proved as a theorem via a more general ranking function definition.

7. Reasoning about General Collusion-Resistant Ranking Functions

We first define certain notations which will be used in the latter analysis:

Definition 1. *T* is an SCG table with columns *W*, *L* and *F* and derived columns *Fault* and *Control*.

Definition 2. T_0 is the empty SCG table.

Definition 3. T_0 is the empty SCG table.

Definition 4. $Win(p)$ is the winning filter for a SCG table given a specified player *p*. More specifically, $T\{Win(p)\}$ will return a sub-table of *T*, where column *W* contains only player *p*.

Definition 5. $Lose(p)$ is the losing filter for a SCG table given a specified player *p*. More specifically, $T\{Lose(p)\}$ will return a sub-table of *T*, where column *L* contains only player *p*.

Definition 6. $Fault(p)$ is the fault filter for a SCG table given a specified player *p*. More specifically, $T\{Fault(p)\}$ will return a sub-table of *T*, where column *Fault* contains only player *p*.

Definition 7. $Control(p)$ is the Controlling filter for a SCG table given a specified player *p*. More specifically, $T\{Control(p)\}$ will return a sub-table of *T*, where column *Control* contains only player *p*.

Definition 8. A ranking function \preceq is an order function between any two players, which takes in a SCG-table as input and returns an order relation between the two players.

We now give a representation theorem for collusion resilient ranking functions. In essence, we show that under basic monotonical properties of ranking functions, the limited collusion effect property is logically equivalent to using a ranking function that is based on a generalized form of fault counting.

A ranking function \preceq is said to be **Local Fault Based (LFB)** if for any two arbitrary players p_x and p_y the relative rank \preceq assigns to p_x with respect to p_y solely depends on the games where p_x or p_y make a fault. Formally,

$$p_x \preceq (T\{Fault(p_x)\} + T\{Fault(p_y)\}) p_y \Leftrightarrow p_x \preceq(T) p_y$$

Theorem 1. *For any ranking function having NNEW and NPEL, CR is equivalent to LFB. Formally, $NNEW \wedge NPEL \Rightarrow (CR \Leftrightarrow LFB)$.*

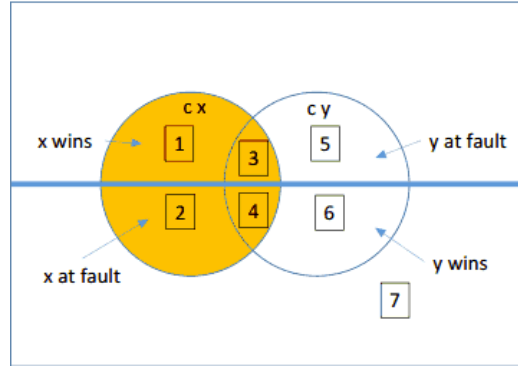


Figure 7: Simplified Venn diagram for all possible game classification in terms of player x and y.

Figure 7 presents a partitioning of the games represented by an arbitrary table T . This partitioning illustrates the intuition behind this theorem and its proof. The intuition is that a ranking function satisfying the LFB property \preceq must completely decide the relative rank of any two arbitrary players p_x and p_y based on the games in the partition 2,4,3,5 only. Games in the other partitions cannot influence the relative rank of p_x and p_y assigned by \preceq .

We now give a proof of this theorem using the partitioning shown in Figure 7. Our theorem follows directly from both lemmas.

Lemma 1. $NNEW \wedge CR \Rightarrow LFB$.

To prove the first lemma, let \preceq be a ranking function that violates the LFB property. By definition of the LFB property, there must be two players p_x and p_y such that the games in regions 2, 4, 3, 5 influence the relative rank assigned by \preceq to p_x and p_y . The influence can either be positive (case I) or negative (case II) for p_x .

Suppose that games in regions 1, 6, 7 positively influence the rank assigned by \preceq to p_x with respect to p_y . But, assuming that \preceq satisfies the CR property, games in regions 1, 7 cannot improve p_x 's rank with respect to p_y because it only contains games not under p_y 's control. Also, assuming that \preceq satisfies the NNEW property, games in region 6 cannot improve p_x 's rank with respect to p_y because it only contains games that p_y has won. Therefore, our assumption that \preceq satisfies both CR and NNEW cannot be true. We have shown the contrapositive for case I.

We now consider case II. Suppose that games in regions 1, 6, 7 negatively influence the rank assigned by \preceq to p_x with respect to p_y . But assuming that \preceq satisfies the CR property, games in regions 6, 7 cannot worsen p_x 's rank with respect to p_y because it only contains games not under p_x 's control. Also, assuming that \preceq satisfies the NNEW property, games in region 1 cannot worsen p_x 's rank with respect to p_y because it only contains games that p_x has won. Therefore, our assumption that \preceq satisfies both CR and NNEW cannot be true. We have shown the contrapositive for case II and the proof of Lemma 1 is now complete.

Lemma 2. $NPEL \wedge LFB \Rightarrow CR$.

To prove the second lemma, let \preceq be a ranking function satisfying both NPEL and LFB. By definition of the LFB property, only games in regions 2, 4, 3, 5 influence the relative rank assigned by \preceq to p_x and p_y . Games in regions 2, 3 and 4 are under the control of p_x . Only games in region 5 can influence the relative rank assigned by \preceq to p_x and p_y , yet games in region 5 are not under the control of p_x . However, games in region 5 are all faults made by p_y and by NPEL they cannot improve the rank of p_y with respect to p_x . Therefore, only games under the control of p_x may worsen p_x 's rank with respect to p_y . A symmetric argument applies to the rank of p_y . This completes the proof of Lemma 2 and hence the theorem.

Definition 9. *A player is perfect if it never makes a fault.*

Next we prove that a perfect player will be ranked at the top provided NNEW, NPEL and CR hold.

Corollary 1. *All perfect players are ranked at the top given a ranking function which satisfies NNEW, NPEL and CR.*

The proof follows directly from Theorem 1. We know that for a ranking function which is NNEW and NPEL, CR is equivalent with LFB. That means only fault counting will effect the ranking result. Also, we know that, a player can't improve his rank by losing a game, and a player can't decrease his rank by winning a game. That means only making a fault results in a lower rank. Hence all perfect players, who make no fault, shall be ranked at the top.

8. Side-Choosing Games for Ranking Functions

We argue in a later section that side-choosing games are applicable in any formal science. The study of ranking functions for side-choosing games is such a formal science. We use semantic games, a special case of side-choosing games, to illustrate the self application.

P is the set of participants.

$$\forall T \in \text{SCG} - \text{Table}(P) \exists r \in \text{Ranking}(P) : r \text{ satisfies NNEW, NPEL, CR and}$$

9. Specific Ranking Functions

9.1 Trivial ranking functions and Fault Counting Ranking Function

We want to instantiate the general theory with specific ranking functions. The simplest one is the trivial ranking function which assigns the same rank to all players. We shall see that the trivial ranking function satisfies NNEW, NPEL and CR.

The proof is simple, while NNEW, NPEL or CR all represented certain property that under some situations, the rank relation among the two specified players should not be effected, and since all players have been assigned the same rank, additional games won't effect the identity among the players. Therefore the trivial ranking function satisfies those properties.

Definition 10. A ranking function is a trivial ranking function \preceq_0 , if and only if, for $\forall T$ and $\forall p_x, p_y$, it always have $p_x \preceq_0(T) p_y$.

Next we introduce the simplest nontrivial structured LFB ranking function: the Fault Counting Ranking Function \preceq_f . We shall see that this function also satisfies NNEW, NPEL and CR. First we introduce the fault counting function and fault counting ranking function:

Definition 11. A fault counting function $faults^T(p)$ for a specified SCG-table T and player p is defined as:

$$faults^T(p) = |T\{Fault(p)\}|$$

Definition 12. A fault counting ranking function \preceq_f for a specified SCG-table T is defined as:

$$p_x \preceq_f(T) p_y \Leftrightarrow faults^T(p_x) \leq faults^T(p_y)$$

The fault counting ranking function should be NNEW and NPEL. Since, intuitively, a win for participant p_x cannot be a fault and therefore cannot increase the number of faults p_x incurs and hence cannot worsen p_x 's rank. A loss for participant p_x may be a fault if p_x was not forced. In this case, the fault count of p_x increases and consequently the rank of p_x may only worsen. A loss for participant p_x while p_x is forced would not be counted and the rank of p_x will not change.

Theorem 2. \preceq_f satisfies the NNEW and NPEL properties.

Furthermore, the fault counting ranking function satisfy CR property. The proof of Theorem 3 falls immediately from Theorem 1 and Lemma 3 which states that the fault counting ranking function has the LFB property.

Theorem 3. \preceq_f satisfies the LCE property.

Lemma 3. \preceq_f satisfies the LFB property

Proof.

Consider the L.H.S of LFB applied to \preceq_f

$$\text{L.H.S.} = p_x \preceq_f(T\{Fault(p_x)\} + T\{Fault(p_y)\}) p_y$$

By definition 18:

$$\Leftrightarrow faults^{T\{Fault(p_x)\}+T\{Fault(p_y)\}}(p_x) \leq faults^{T\{Fault(p_x)\}+T\{Fault(p_y)\}}(p_y)$$

By definition 17:

$$\Leftrightarrow faults^T(p_x) \leq faults^T(p_y)$$

By definition 18:

$$\Leftrightarrow p_x \preceq_f(T) p_y = \text{R.H.S.}$$

□

9.2 Weighted Fault Counting Ranking Function

Based on the simplest Fault Counting Function in the last section, we can further present a family of collusion-resistant ranking functions which measure different kinds of weaknesses of the participants.

We consider positive weights assignment to fault classes where:

1. The winner is a forced opponent, weakness in proponent: Weight = α .
2. The winner is a forced proponent, weakness in opponent: Weight = β .
3. The winner is a non-forced opponent, weakness in proponent: Weight = γ
4. The winner is a non-forced proponent, weakness in opponent: Weight = δ

Then we can define a serial of new filters for SCG-Table T in regards of those weights:

Definition 13. $Fault_\alpha(p)$ is the fault filter for a SCG table given a specified player p . More specifically, $T\{Fault(p)\}$ will return a sub-table of T , where column Fault contains only player p who made fault in the first class.

Definition 14. $Fault_\beta(p)$ is the fault filter for a SCG table given a specified player p . More specifically, $T\{Fault(p)\}$ will return a sub-table of T , where column Fault contains only player p who made fault in the second class.

Definition 15. $Fault_\gamma(p)$ is the fault filter for a SCG table given a specified player p . More specifically, $T\{Fault(p)\}$ will return a sub-table of T , where column Fault contains only player p who made fault in the third class.

Definition 16. $Fault_\delta(p)$ is the fault filter for a SCG table given a specified player p . More specifically, $T\{Fault(p)\}$ will return a sub-table of T , where column Fault contains only player p who made fault in the fourth class.

With definitions of those new filters, we can now give a definition of Weighted Fault Counting Ranking Function as following:

Definition 17. A weighted fault counting function $faults_\lambda^T(p)$ for a specified SCG-table T and player p is defined as:

$$faults_\lambda^T(p) = \lambda |T\{Fault_\lambda(p)\}|, \lambda \in \{\alpha, \beta, \gamma, \delta\}$$

Definition 18. A Weighted Fault Counting Ranking Function \preceq_w for a specified SCG-table T is defined as:

$$p_x \preceq_w(T) p_y \Leftrightarrow \sum_{\lambda \in \{\alpha, \beta, \gamma, \delta\}} faults_\lambda^T(p_x) \leq \sum_{\lambda \in \{\alpha, \beta, \gamma, \delta\}} faults_\lambda^T(p_y)$$

$$U(D) = \alpha \cdot \alpha Rows + \beta \cdot \beta Rows + \gamma \cdot \gamma Rows + \delta \cdot \delta Rows$$

By suitably choosing the weights, we can influence the competitiveness of the games. For example, if $\beta = \gamma = \delta = 0$ and $\alpha > 0$ there is less competition because three kinds of weaknesses are not counted. And notice that once we assign identical value to all weights, the weighted fault counting function will degenerate to a simple fault counting function.

Following the proof of simple fault counting ranking function in the last section. It's easy to show that the weighted fault counting function is also NNEW, NPEL and CR.

10. Applications of Side-Choosing Games

Our entry point to side-choosing games were semantic games which have been studied by logicians for a long time [SEP]. Give definition. Give simplified semantic games: one value for each quantified variable.

10.1 Meritocracy Evaluation

One important application for side-choosing games is meritocracy evaluation. Since we know that the less faults a player has made, the higher one should be ranked, so the one (we call such a player "quasi-perfect", because the claim he or she defended might actually be refuted by those who are stronger but not participated in the games) who made no faults should be top-ranked. Formally, we have the following theorem:

Meritocracy Theorem. *NNEW and NPEL and CR imply that a quasi-perfect player is top-ranked, where the quasi-perfect player is a player who never makes a fault.*

10.2 Teaching Formal Sciences

Side-choosing games are attractive for teaching staffs in formal sciences for the following reasons:

- Students become active participants in teaching

- students accept that when they make a fault there is a problem in their argument: They chose a side but could not defend it. This holds independent of whether they chose the proponent or opponent role.

- collaboration with a competitive flair

- can use teaching environments like Piazza (with disadvantage that all communications are open within each team). Use delayed publication.

10.2.1 TEACHING ALGORITHMS

10.2.2 TEACHING SOFTWARE DEVELOPMENT

10.3 Organizing Problem Solving Communities for Constructive Formal Sciences

10.3.1 ALGORITHM DEVELOPMENT

10.3.2 SOFTWARE DEVELOPMENT

10.3.3 ROBOTS

10.3.4 DIALOG-BASED PROOFS

11. Future work

11.1 Extending SCG-Tables

When more information is known about games and their results we can extend the SCG table. For example, the current SCG-table does not include information on whether the claim is true or false. With this information added, we can consider new ranking functions.

- From old paper

12. Further Applications of Side-Choosing Games

Wikipedia of Computations

OLD123 =====

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 proponent. The semantic game takes place between the proponent and opponent of the claim and requires the players to compute objects that defend the players’ positions of being proponent or opponent. 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.

12.0.1 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 proponent (if the avatar thinks the claim is true) or opponent (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 proponents v_1, v_2 and 2 opponents f_1, f_2 , the CIM will create 2 games: v_1/f_1 and v_2/f_2 and not two games v_1/v_2 and f_1/f_2 .

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,

$$\begin{aligned} \text{MinBasisSize}(G \in \text{Digraphs}, n \in \mathbb{N}) = \\ \text{BasisSize}(G, n) \wedge \forall k \text{ s.t. } k < n : \neg \text{BasisSize}(G, k) \end{aligned}$$

where

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

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 proponent or opponent. Let's assume, the avatar wants to be proponent and the CIM has found another avatar who wants to be opponent. Who is right? The semantic game between the two will give us further information:

There are two cases:

- First conjunct

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

- Second conjunct

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

The above claim $\text{BasisSize}(G, n)$ 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 $\text{reachAll}(G, s)$ 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 proponent and opponent:

$$BasisSize(G \in Digraphs, n \in \mathbb{N}) = \\ \exists s \in \mathcal{P}(nodes(G)) \text{ s.t. } |s| = n : reachAll(G, s)$$

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:

$$BasisSize(G \in DAGs, n \in \mathbb{N}) = \\ \exists s \in \mathcal{P}(nodes(G)) \text{ s.t. } |s| = n : reachAll(G, s).$$

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: $\forall G_1 : \text{if } BasisSize(G_1, n) \text{ then } BasisSize(f(G_1), n)$.

12.1 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 (Marion, 2009), (Hodges, 2009), (Keiff, 2011).

SGs are played between two players, the *proponent* and the *opponent*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 (Pietarinen, 2000). The existence of a winning strategy for the *proponent* implies that the underlying logical statement is indeed *true* and the existence of a winning strategy for the *opponent* implies that the underlying logical statement is indeed *false*.

Players need to solve *computational problems* in the course of playing SGs. For example, the opponent of the $prime(7) = \forall k \text{ s.t. } 1 < k < 7 : \neg divides(k, 7)$ 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

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

logically specified using the claim $\forall n \exists s : \forall k : \text{divides}(k, n) \Leftrightarrow k \in s$. In SGs derived from this claim, the proponent needs to correctly solve $\text{factors}(n)$ 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.

13. Thesis

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

13.1 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 proponent position and a user to take on the opponent 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.

13.1.1 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 proponent position and the player taking the opponent position. For example, the tuple $\langle c, p_1, p_2 \rangle$ represents a state where p_1 is the player taking the proponent position and p_2 is the player taking the opponent 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 13.2).

13.1.2 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 (Joglekar, Garcia-Molina, & Parameswaran, 2012).

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.

13.1.3 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.

13.1.4 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 proponent 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.

13.1.5 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

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 (Pink, 2011), (Kahneman, 2011), (Ipeirotis & Paritosh, 2011).

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 (Doan, Ramakrishnan, & Halevy, 2011).

13.2 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.

13.2.1 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.

13.2.2 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.

13.2.3 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 (systems on the World-Wide Web,). 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.

14. Initial Investigation

To support our thesis, we designed and partially implemented (The polymath blog,) a proof of concept SG-based crowdsourcing system. Our system constitutes a redesign from scratch of the Scientific Community Game (SCG) (Abdelmeged & Lieberherr, 2013), (Abdelmeged & Lieberherr, 2012), (Lieberherr, Abdelmeged, & Chadwick, 2010) 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.

14.1 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 (Kulas & Hintikka, 1983) (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 14.2. 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 14.3. To estimate crowd beliefs, our system uses a simple formula that is presented in Section 14.4. 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.

14.2 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 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 3 which summarizes CAGs. The columns of the table indicate the name of the SG being played, the

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 3: The Contradiction-Agreement Game

forced scholar (if any), the SG winner, and whether the contribution of the winner is potentially correct (assuming that “cheating” is somehow discouraged).

14.2.1 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.

14.3 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 $Payoff(S_1, S_2)$. The strength of user S is denoted by $Str(S)$. The algorithm is given by:

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

Ideally, we would like the strengths produced by the algorithm to be *consistent* with the payoffs (i.e. $\forall S_1, S_2 : \text{Payoff}(S_1, S_2) \geq \text{Payoff}(S_2, S_1) \Rightarrow \text{Str}(S_1) \geq \text{Str}(S_2)$). However, the relation $R(S_1, S_2) = \text{Payoff}(S_1, S_2) \geq \text{Payoff}(S_2, S_1)$ is not necessarily transitive while the relation $Q(S_1, S_2) = \text{Str}(S_1) \geq \text{Str}(S_2)$ 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. $\forall i, j \text{Payoff}(S_i, S_j) > \text{Payoff}(S_j, S_i) \wedge \forall k \neq i, j : \text{Payoff}(S_i, S_k) \geq \text{Payoff}(S_j, S_k) \wedge \text{Payoff}(S_j, S_k) \leq \text{Payoff}(S_i, S_k) \Rightarrow \text{Str}(S_i) \geq \text{Str}(S_j)$.
2. A scholar that only won(lost) games will have a strength of 1(0). Formally, $\forall i \forall j \text{Payoff}(S_i, S_j) = 0 \wedge \exists j \text{Payoff}(S_j, S_i) > 0 \Rightarrow \text{Str}(S_i) = 0$, and $\forall i \forall j \text{Payoff}(S_j, S_i) = 0 \wedge \exists j \text{Payoff}(S_i, S_j) > 0 \Rightarrow \text{Str}(S_i) = 1$. A scholar that has not won or lost any games will have a strength of 0.5. Formally, $\forall i \forall j \text{Payoff}(S_j, S_i) = 0 \wedge \text{Payoff}(S_i, S_j) = 0 \Rightarrow \text{Str}(S_i) = 0.5$.

14.4 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 $c_F / (c_T + c_F)$.

15. 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.

15.1 Software Development

The most successful graduate classes were the ones that developed and maintained the software for SCG Court (Abdelmeged & Lieberherr, 2011) 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 (Kleinberg & Tardos, 2005) where the best avatars needed to solve a constrained search problem using a modified Pascal triangle.

15.2 Algorithms

The most successful course (using (Kleinberg & Tardos, 2005) 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 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.

16. Related Work

16.1 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 (Bernstein, Klein, & Malone, 2012). 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 (Attenberg, Ipeirotis, & Provost, 2011).

Scientific discovery games, such as FoldIt and EteRNA, are variants of the SCG. (?) describes the challenges behind developing scientific discovery games. (Andersen, O’Rourke, Liu, Snider, Lowdermilk, Truong, Cooper, & Popovic, 2012) 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.

16.2 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 (Marion, 2009), (Hodges, 2009) have a long history going back to Socrates. The SCG is an imperfect information game which builds on Paul Lorenzen’s dialogical games (Keiff, 2011).

16.3 Foundations of Digital Games

A functioning game should be deep, fair and interesting which requires careful and time-consuming balancing. (Jaffe, Miller, Andersen, Liu, Karlin, & Popovic, 2012) 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.

16.4 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 (Scacchi, 2012) 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).

16.5 Online Judges

An online judge is an online system to test programs in programming contests. A recent entry is (Petit, Giménez, & Roura, 2012) 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.

16.6 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. (Andersen, 2012) 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.

16.7 Formal Sciences and Karl Popper

James Franklin points out in (Franklin, 1994) 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 (Popper, 1969) is the father of non-deductive methods in science. The SCG is a way of doing science on the web according to Karl Popper.

16.8 Scientific Method in CS

Peter Denning defines CS as the science of information processes and their interactions with the world (Denning, 2005). The SCG makes the scientific method easily accessible by expressing the hypotheses as claims. Robert Sedgewick in (Sedgewick, 2010) 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.

16.9 Games and Learning

Kevin Zollman studies the proper arrangement of communities of learners in his dissertation on network epistemology (Zollman, 2007). 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.

16.10 Origins of SCG

A preliminary definition of the SCG was given in a keynote paper (Lieberherr et al., 2010). (Lieberherr, 2009) gives further information on the SCG. The original motivation for the SCG came from the two papers with Ernst Specker: (Lieberherr & Specker, 1981) and the follow-on paper (Lieberherr & Specker, 2012). 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.

17. 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.

17.1 Model Development

17.1.1 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, $MinBasisSize(G \in Digraphs, n \in \mathbb{N}) = BasisSize(G, n) \wedge \forall k \text{ s.t. } k < n : \neg BasisSize(G, k)$ where $BasisSize(G \in Digraphs, n \in \mathbb{N}) = \exists s \in \mathcal{P}(nodes(G)) \text{ s.t. } |s| = n : \forall m \in nodes(G) \exists p \in paths(G) : first(p) = m \wedge last(p) \in s$.
2. *Number of source nodes of a DAG*: a source node is a node with no incoming edges. Formally, $\#src(D \in DAGs, m \in \mathbb{N}) = \exists s \in \mathcal{P}(nodes(D)) \text{ s.t. } |s| = m : \forall v \in nodes(D) : inDegree(v) = 0 \Leftrightarrow v \in s$.

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?

17.1.2 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 opponent to provide a *slightly* better solution to win. As remedy, it is possible to bias the situation towards the proponent by requiring the opponent to provide a solution that is at well better than the solution provided by the proponent 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.

17.2 System Development

We propose to turn the current implementation (Cur,) into a web based application. We also propose to further develop the claim language and the CIM and the encouragement and retention scheme.

17.2.1 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, $AlgoRunTime(c, n_{min}, n_{max}) = \exists a \in Algo : \forall i \in Input \text{ s.t. } n_{min} \leq size(i) \leq n_{max} : correct(i, a(i)) \wedge RunTime(a(i)) \leq c * size(i)$.
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 (Väänänen, 2007).
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: $\text{sat}(f) = \max_J \text{csat}(f, J)$, instead of: $\text{sat}(f, x) = \exists J \text{ s.t. } \text{csat}(f, J) = x : \forall H : \text{csat}(f, H) \leq \text{csat}(f, J)$

4. add an abstraction facility.

17.2.2 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 (Hodges, 2009) where in each round, one player 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 be 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 positions 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.

17.2.3 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.

17.3 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 (Logic:,).

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

References

- Website. <https://github.com/amohsen/fscp>.
- Algorithm development through crowdsourcing.. <http://catalyst.harvard.edu/services/crowdsourcing/algosample.html>.
- The polymath blog. Website. <http://polymathprojects.org/>.
- Abdelmeged, A., & Lieberherr, K. J. (2011). SCG Court: Generator of teaching/innovation labs on the web. Website. <http://sourceforge.net/p/generic-scg/code-0/110/tree/GenericSCG/>.
- Abdelmeged, A., & Lieberherr, K. J. (2012). The Scientific Community Game. In *CCIS Technical Report NU-CCIS-2012-19*. <http://www.ccs.neu.edu/home/lieber/papers/SCG-definition/SCG-definition-NU-CCIS-2012.pdf>.
- Abdelmeged, A., & Lieberherr, K. J. (2013). FSCP: A Platform for Crowdsourcing Formal Science. In *CCIS Technical Report*. http://www.ccs.neu.edu/home/lieber/papers/SCG-crowdsourcing/websci2013_submission_FSCP.pdf.
- Andersen, E. (2012). Optimizing adaptivity in educational games. In *Proceedings of the International Conference on the Foundations of Digital Games, FDG '12*, pp. 279–281, New York, NY, USA. ACM.
- Andersen, E., O'Rourke, E., Liu, Y.-E., Snider, R., Lowdermilk, J., Truong, D., Cooper, S., & Popovic, Z. (2012). The impact of tutorials on games of varying complexity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '12*, pp. 59–68, New York, NY, USA. ACM.

- Attenberg, J., Ipeirotis, P., & Provost, F. (2011). Beat the machine: Challenging workers to find the unknown unknowns. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*.
- Beasley, J. (2006). *The Mathematics of Games*. Dover books on mathematics. Dover Publications, Incorporated.
- Bernstein, A., Klein, M., & Malone, T. W. (2012). Programming the global brain. *Commun. ACM*, 55(5), 41–43.
- Chadwick, B. (2008). DemeterF: The functional adaptive programming library. Website. <http://www.ccs.neu.edu/home/chadwick/demeterf/>.
- Denning, P. J. (2005). Is computer science science?. *Commun. ACM*, 48(4), 27–31.
- Doan, A., Ramakrishnan, R., & Halevy, A. Y. (2011). Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4), 86–96.
- Elo, A. (1978). *The rating of chessplayers, past and present*. Arco Pub.
- Franklin, J. (1994). The formal sciences discover the philosophers’ stone. *Studies in History and Philosophy of Science*, 25(4), 513–533.
- Gonzalez-Daz, J., Hendrickx, R., & Lohmann, E. (2014). Paired comparisons analysis: an axiomatic approach to ranking methods. *Social Choice and Welfare*, 42(1), 139–169.
- Hodges, W. (2009). Logic and games. In Zalta, E. N. (Ed.), *The Stanford Encyclopedia of Philosophy* (Spring 2009 edition).
- Ipeirotis, P. G., & Paritosh, P. K. (2011). Managing crowdsourced human computation: a tutorial. In *Proceedings of the 20th international conference companion on World wide web, WWW ’11*, pp. 287–288, New York, NY, USA. ACM.
- Jaffe, A., Miller, A., Andersen, E., Liu, Y.-E., Karlin, A., & Popovic, Z. (2012). Evaluating competitive game balance with restricted play..
- Joglekar, M., Garcia-Molina, H., & Parameswaran, A. (2012). Evaluating the crowd with confidence. Technical report, Stanford University.
- Kahneman, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.
- Keiff, L. (2011). Dialogical logic. In Zalta, E. N. (Ed.), *The Stanford Encyclopedia of Philosophy* (Summer 2011 edition).
- Kleinberg, J., & Tardos, E. (2005). *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Kulas, J., & Hintikka, J. (1983). *The Game of Language: Studies in Game-Theoretical Semantics and Its Applications*. Synthese Language Library. Springer.
- Langville, A., & Meyer, C. (2012). *Who’s #1?: The Science of Rating and Ranking*. Princeton University Press.
- Lieberherr, K. (2009). The Scientific Community Game. Website. <http://www.ccs.neu.edu/home/lieber/evergreen/specker/scg-home.html>.
- Lieberherr, K. J., Abdelmeged, A., & Chadwick, B. (2010). 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*. <http://www.ccs.neu.edu/home/lieber/evergreen/specker/paper/bionetics-2010.pdf>.
- Lieberherr, K. J., & Specker, E. (1981). Complexity of Partial Satisfaction. *Journal of the ACM*, 28(2), 411–421.
- Lieberherr, K. J., & Specker, E. (2012). Complexity of Partial Satisfaction II. *Elemente der Mathematik*, 67(3), 134–150. <http://www.ccs.neu.edu/home/lieber/p-optimal/partial-sat-II/Partial-SAT2.pdf>.
- Marion, M. (2009). Why Play Logical Games. Website. <http://www.philomath.uqam.ca/doc/LogicalGames.pdf>.
- Petit, J., Giménez, O., & Roura, S. (2012). Jutge.org: an educational programming judge. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education, SIGCSE '12*, pp. 445–450, New York, NY, USA. ACM.
- Pietarinen, A. (2000). Games as formal tools vs. games as explanations. Tech. rep..
- Pink, D. (2011). *Drive: The Surprising Truth About What Motivates Us*. Canongate Books.
- Popper, K. R. (1969). *Conjectures and refutations: the growth of scientific knowledge*, by Karl R. Popper. Routledge, London.
- Rubinstein, A. (1980). Ranking the participants in a tournament. *SIAM Journal on Applied Mathematics*, 38(1), pp. 108–111.
- Scacchi, W. (2012). The Future of Research in Computer Games and Virtual Worlds: Workshop Report. Technical Report UCI-ISR-12-8. http://www.isr.uci.edu/tech_reports/UCI-ISR-12-8.pdf.
- Sedgewick, R. (2010). The Role of the Scientific Method in Programming. Website. <http://www.cs.princeton.edu/rs/talks/ScienceCS.pdf>.
- Tulenheimo, T. (2013). Independence friendly logic. In Zalta, E. N. (Ed.), *The Stanford Encyclopedia of Philosophy* (Fall 2013 edition).
- Väänänen, J. (2007). *Dependence Logic*. London Mathematical Society Student Texts. Cambridge University Press.
- Zollman, K. J. S. (2007). The communication structure of epistemic communities. *Philosophy of Science*, 74(5), 574–587.

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 8 shows the grammar for a formula expressed using the class dictionary notation (Chadwick, 2008).

```

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 8: Formula Language

B. Semantic Games

Given a claim c and two scholars, a proponent *ver* and an opponent *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 $\phi = R(t_1, \dots, t_n)$ and $M, g \models R(t_1, \dots, t_n)$, *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 (Kulas & Hintikka, 1983), (Tulenheimo, 2013). We slightly modified Hintikka's original definition to handle the quantification predicate in our language.