Computational Problem Solving Using Generalized Semantic Games

A dissertation presented

by

Ahmed Abdelmeged

to the Faculty of the Graduate School

of the College of Computer and Information Science

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Northeastern University

Boston, Massachusetts

September, 2013

ABSTRACT

Competitions have been used to drive scientific research and innovation on open problems. Currently, most mainstream competitions organized to drive scientific research follow the contest pattern where the performance of research products is measured against a specific set of static benchmarks with no direct communication between participants in the context of the competition.

We believe that Semantic Games (SGs) of interpreted logic sentences provide a useful foundation to organize scientific research competitions. A Semantic Game (SG) is a two-party debate of the truth of a particular interpreted logic sentence where participants are required to solve computational problems to choose and defend their positions in the debate. The debate is structured as a two-person game. We believe SGs are useful because their participants also collaborate in the sense that the winner of a semantic game must point out to a mistake made by their opponent.

This dissertation describes our research regarding the development and evaluation of the Scientific Community Game (SCG) as an alternative approach to organize scientific research competitions for solving computational problems. SCG is a sports-like tournament of semantic games, guaranteeing that a perfect participant must rank at the top.

The main challenge in developing SCG is that standard tournament formats, such as elimination, round robin, and Swiss, cannot be used with SGs because SGs can only be played between two participants *voluntarily* taking opposite sides. To adopt SGs to fit in standard tournament formats, it is inevitable to force a side on one of the participants in an SG when both participants choose the same side. Forcing a side on a randomly chosen participant creates a game with imbalanced winning chances. Consequently, there is no guarantee that a perfect participant will rank on top.

The main challenge in evaluating SCG is that a formal characterization of the best participant in a given set of participants is needed. This can be done in two different ways, using a reference tournament or using a model of the performance of a participant of a given strength in an SG. Unfortunately, there is no clear objective way to decide on a reference tournament. Furthermore, existing models for participants' performance based on their strength such as the Thurstone-Mosteller model [42] and the Bradley-Terry model [13] where developed for sports games and their underlying assumptions are not preserved by SGs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Competitions have been used to drive research and innovation on open problems. Competitions motivate their egoistic participants to invest their effort to gain recognition, and possibly a prize too, for winning the competition. The uncertainty, and sometimes the prize, of a competition can draw the attention of spectators and help building a community around the open problem.

More importantly, a competition reveals some meta-knowledge about its participants' knowledge of the domain of the open problem. For example, one may assume that the winner of a competition has the best knowledge about the domain of the open problem and is thus the closest to solve it, provided that the rules of the competition are reasonably fair. Depending on the rules, some competitions may help diffusing knowledge of the domain of the open problem.

There are several examples of competitions, historical and modern, as well as platforms that facilitate the organization of competitions with ad-hoc rules. Examples of historical competitions include the contest held, in 1535, between Tartaglia and Fior to discover who knows how to solve cubic equations more efficiently. The rules were that each provides the other with 30 cubic equations to solve and the faster wins. Examples of modern competitions include the SAT competition [6] or-

ganized to discover the best performing SAT solver on a specific set of benchmarks, and the Netflix prize competition [3] organized to discover the best performing prediction algorithm on a specific dataset. Examples of platforms that facilitate competitions include TopCoder [43] which facilitates the organization of competitions to develop software artifacts, Kaggle [2] which facilitates the organization of data mining competitions, Foldit [15] which organizes protein foldings competitions, EteRNA [7] which organizes protein synthesis competitions, and Project Euler [5] as well as Jutge [32] which organize competitions for educational purposes.

With few exceptions, mainstream competitions organized to drive scientific research follow a specific pattern; the contest pattern where the performance of research products is measured against a specific set of static benchmarks. As a consequence, communications received by participants in the context of a competition originate in a central administrator organizing the competition according to the rules. Participants do not directly communicate in the context of the competition.

In some competitions, participants may influence benchmarks which may be considered as an indirect, one-way communication between participants. In the SAT competition, there is an informal process by which a participant may influence the benchmarks; a call for benchmarks is sent out before the competition. The competition owners choose which of the submitted benchmarks to include in the competition. In Foldit, a central administrator keeps a database of best known protein foldings to compare foldings submitted by participants in the context of a competition against the protein foldings database. Protein foldings submitted by participants are automatically integrated into the database of best known protein foldings.

The TopCoder platform enables a restricted form of direct communication between developers participating in a competition; developers get a brief chance to submit an optional test input to their opponents' programs. The point is to test

corner cases where a particular input can cause the opponents' programs to crash or run for too long. Developers do not submit test cases, consisting of inputs and expected outputs, as it would enable the unproductive behavior of submitting test cases with the wrong expected output. Furthermore, developers are discouraged from submitting the optional test input as they are penalized in case their test input does not cause any of the opponents' programs to crash or to run for too long.

In this dissertation, we present an alternative approach to organize a scientific community around a formally specified computational problem. Nowadays, formally specified computational problems are becoming commonplace in natural sciences such as physics and biology [24] (See Section 2.2 for more details). Our approach is based on direct two-way communication between participants. We start by presenting a thesis summarizing our approach, then we present a list of our contributions, then we describe three application areas, and finally we describe the organization of this dissertation.

1.1 Thesis

Our thesis is: "Semantic games of interpreted logic sentences provide a useful foundation to collectively solve formally specified computational problems with several unreliable participants.".

A Semantic Game (SG) is a two-party debate of the truth of a particular interpreted logic sentence, or claim for short. The debate is structured as a two-person game with two distinguished participants, the verifier and the falsifier. The party taking the position that the underlying claim is $true$ (respectively $false$) becomes the verifier (respectively falsifier). The rules of the game are derived from the syntax of the underlying claim such that the game gives the claim a meaning in the sense that if there is a winning strategy for the verifier (respectively falsifier) then

the underlying claim is $true$ (respectively $false$).

For example, consider the claim "x is $o(x^2)$". This claim can be formally defined as $\forall c > 0 : \exists x_0 \forall x \geq x_0 : |x| \leq c \cdot |x^2|$ interpreted in the structure of real arithmetic. An SG for this claim proceeds as follows:

1. The falsifier provides a value for c that makes the, logically weaker, subclaim $\exists x_0 \forall x \geq x_0 : |x| \leq c \cdot |x^2|$ false.

2. Given the value provided by the falsifier for c, the verifier provides a value for x_0 that makes the, logically stronger, subclaim $\forall x \geq x_0 : |x| \leq c \cdot |x^2|$ true.

3. Given the values for c and x_0 provided so far, the falsifier provides a value for x that is greater than or equal to x_0 and makes the, logically weaker, subclaim $|x| \leq c \cdot |x^2|$ false.

To play an SG, participants must initially decide the underlying claim in order to select a side in the debate. Then, depending on the underlying claim and the side they select, participants are required to solve certain computational problems. For example finding c, finding x_0 given c, finding x given c and x_0 in the aforementioned example.

The losing party in an SG can always be blamed for falling into a contradiction in the following sense: suppose that the losing party is the verifier (respectively falsifier), then either the verifier (respectively falsifier) has chosen the wrong side, or the verifier (respectively falsifier) must have strengthened (weakened) the underlying claim too much to $false$ (respectively $true$) by failing to correctly solve a required computational problem. There are no ties in SGs even if neither party correctly solves all required computational problems.

SGs can be used to combine the efforts of two participants to collectively solve the required computational problems of a given claim in the following two senses:

1. In the short term, competitive, sense, an SG provides an objective basis to discriminate between participants and their solutions to the required computational problems. Winning participant's solutions can be assumed to be the better solution.Also, the winning participant can be assumed to have a better grasp of the underlying structure.

2. In the long term, collaborative, sense, to win an SG, the winning participant must point out a specific mistake in the losing participant's solutions. Assuming that SGs are repetitively played in the context of some process (such as a research, development or education process) and that participants are keen on winning and can effectively learn from their mistakes, the reliability of the winner's solution improves over time [14](Section 2.3 gives more details about embedding SGs in a process).

To support our thesis, we developed the Scientific Community Game (SCG); SCG is a tournament of SGs, guaranteeing that a perfect participant must rank at the top. The main challenge in developing SCG is that standard tournament formats, such as elimination, round robin, and Swiss, cannot be used with SGs because SGs can only be played between two participants *voluntarily* taking opposite sides. To adopt SGs to fit in standard tournament formats, it is inevitable to force a side on one of the participants in an SG when both participants choose the same side. Forcing a side on a randomly chosen participant creates a game with imbalanced winning chances. Consequently, there is no guarantee that a perfect participant will rank on top.

There is another challenge in evaluating different tournament design alternatives for SCG. In order to evaluate a tournament design alternative , a formal characterization of the best participant in a given set of participants is needed. This can be done in two different ways, using a reference tournament or using a model

of the performance of a participant of a given strength in an SG. Unfortunately, there is no clear objective way to decide on a reference tournament. Furthermore, existing models for participants' performance based on their strength such as the Thurstone-Mosteller model [42] and the Bradley-Terry model [13] where developed for sports games and their underlying assumptions are not preserved by SGs.

1.2 Contributions

This research is about designing, implementing, and empirically evaluating SCG. More precisely, we claim the following contributions:

1. We propose the idea of organizing a scientific community around an SG-based competition, called SCG.

2. We present a number of alternative SCG designs based on adopting SGs to standard tournament formats.

3. We present a latent variable model relating the latent strength of participants to the winning probabilities of participants in an SG.

4. We empirically evaluate SCG design alternatives we presented.

5. We report on the Formal Science Wikipedia(FSW) [8]; an SCG implementation intended to organize formal scientific knowledge in an active, disputable form on the web.

1.3 SCG Applications

SCG can be directly applied to the following areas:

1. Crowdsourcing solutions for computational problems: computational problems such as decision, search, optimization, counting, and promise problems can be formulated as decision problems.

2. Software development: as a software development process for components solving computational problems.

3. Education: gives a minimal structure to Self Organized Learning Environments (SOLEs) [18] for formal knowledge. We have already used SGs in teaching an algorithms class in the spring of 2012 using Piazza [4] as a communication medium. Competition encouraged students to invest time to solve algorithmic problems such as the problem of finding the worst-case inputs for the Gale-Shapely stable matching algorithm. We also observed that a few good students became effective teachers for the rest.

1.4 Organization

The rest of this dissertation is organized as follows: In Chapter 2 we present background information on SGs and how they can be utilized to organize communities around computational problems. In Chapter 3 we describe the related work to SCG. More specifically, we describe work related to our choice of SGs, to balancing the winning chances in games, to tournament design and to the SCG applications. In Chapter 4 we present a number of alternative tournament designs for SCG. In Chapter 5 we present a model for participants' performance based on their strength. We also present an empirical study aiming at selecting the best tournament design for SCG. In Chapter 6 we present the Formal Science Wikipedia, an SCG implementation intended to organize formal scientific knowledge in an active, disputable form on the web. Chapter 7 concludes this dissertation.

Chapter 2

Background

In this chapter we present background information on SGs and on our approach of utilizing SGs to organize communities around computational problems. In Section 2.1, we describe SGs, their rules and how are they are played. Then, in Section 2.2, we describe our approach to formulating claims. More specifically, how to leverage the slightly restricted interpretation structure of claims to 1) formulate complex computational problems arising in natural sciences, and to 2) focus the community on important computational problems. Finally, in Section 2.3 we describe a workflow for a community to collaborate on solving a computational problem via SGs.

2.1 Semantic Games

A claim is a logical sentence interpreted in a computable structure. A claim family is a parameterized claim. An SG is a formal two-party debate of the *truth*[1] of a particular claim. The two sides of the debate are called the *verifier* side and the

[1] as opposed to logical validity.

falsifier side [2]; participants taking the verifier side assert that the claim is true. s taking the falsifier side assert that the claim is false.

In the theory of SGs, claims derive their meaning from SGs [33]; the existence of a winning strategy for the *verifier* implies that the underlying claim is indeed *true* and the existence of a winning strategy for the *falsifier* implies that the underlying logical sentence is indeed *false*.

The rules of SGs are prompted by the logical connectives encountered in claims. Table 2.1 shows the rules for first order logic proposed by Hintikka [29]. We use $SG(\langle \Phi, A \rangle, ver, fal)$ to denote an SG where the underlying claim is comprised of the formula Φ interpreted in the structure A and ver, respectively fal, denotes the participant taking the verifier, respectively falsifier, side. For universally quantified formulas $\forall x : \Psi(x)$, the falsifier provides a value x_0 for the quantified variable x and the game proceeds as $SG(\langle \Psi[x_0/x], A \rangle, ver, fal)$. For existentially quantified formulas, the verifier provides a value for the quantified variable. For and-compounded formulas, the falsifier chooses one of the subformulas. For or-compounded formulas, the verifier selects a subformula. For negated formulas $\neg\Psi$, no moves are required and the game proceeds as $SG(\langle \Psi, A \rangle, fal, ver)$. Primitive formulas are evaluated in the underlying structure and the verifier wins if they hold, otherwise the falsifier wins.

2.1.1 Playing Semantic Games

In an SG, participants are required to:

1. Take a side on the underlying claim. They do so by solving an instance of a decision problem.

[2] other names have been also used in the literature such as *I* and *Nature*, *Proponent* and *Opponent*, *Alice* (female) and *Bob* (male), and ∃loise and ∀belard.

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

Table 2.1: moves for $SG(\langle \Phi, A\rangle, ver, fal)$

2. Support their side. They do so by solving several other computational problem instances depending on the syntax of the underlying claim.

We illustrate this point through the claim $SP(0.5)$ from the claim family defined by the formula $SP(c \in [0,1]) := \forall x \in [0,1] \exists y \in [0,1] : x \cdot y + (1-x) \cdot (1-y^2) \geq c$ interpreted in the structure of real arithmetic. First, participants decide whether $SP(0.5)$ holds. s that decide that $SP(0.5)$ holds become verifiers. s that decide that $SP(0.5)$ does not hold become falsifiers. Let fal be an arbitrary falsifier, and *ver* be an arbitrary verifier. According to the rules, fal is required to provide a value for the universally quantified variable x. Suppose that fal provided 0 for x. The game proceeds as a game between the same two participants taking the same sides but with the claim $\exists y \in [0,1] : (1-y^2) \geq 0.5$. According to the rules, *ver* is required to provide a value for existentially quantified variables. Suppose that *ver* provided 0, then the game proceeds on the claim $1 \geq 0.5$. This is a true primitive claim, according to the structure of real arithmetic, and therefore the *ver* wins.

2.1.2 Avatars

An avatar represents a human participant in a computer game. An SG avatar provides a solution for the computational problems of the underlying claim. It is certainly desirable to have fully automated avatars. It is also possible ensure that an SG

avatar is fully automated by sandbox execution. However, there are claims where it is very difficult or even impossible to develop a fully automated avatar.

2.2 Formulating Claims

A logical specification of a computational problem p constitutes a claim family CF such that at least one of the participants would be required to solve an instance of p while playing an SG with an underlying claim from the claim family CF. In the following paragraphs we show examples of claim families specifying different kinds of computational problems.

A decision problem can be specified by an arbitrary claim family. For example, consider Bertrand's postulate that there is always at least one prime between n and $2 \cdot n$ which can be specified as $Bertrand() := \forall n \in \mathbb{N} \setminus \{0,1\} : \exists k \; s.t. \; k \in \mathbb{N} \wedge n \leq k \leq 2 \cdot n : prime(k)$. In an SG with $Bertrand()$ as the underlying claim, participants would be required to solve the following computational problems:

- deciding Bertrand's postulate,
- searching for an n making $\exists k \in [n, 2 \cdot n] : prime(k)$ false, and
- given and n, searching for a $k \in [n, 2 \cdot n]$ such that $prime(k)$ holds.

A search or function problem can be specified by a singleton claim family of the form $\forall i \; : \exists o : \Phi(i,o)$ where Φ is a logical formula that holds when o is the correct output for the input i. For example, consider the search problem of finding a topological ordering of a DAG. This problem can be specified as $TopologicalOrdering() := \forall g \in DAG : \exists s \in sequences(nodes(g)) : correcTopologicalOrdering(g,s)$.

An optimization problem can be specified by a singleton claim family of the form $\forall i : \exists o_1 : \forall o_2 : \Psi(i,o_1,o_2)$ where $\Psi(i,o_1,o_2)$ is a logical formula that holds

when o_1 is the correct output for i and o_1 is better than o_2 or when o_2 is not a correct output for i. For example, the MAX-SAT problem can be specified as $\forall f \in CNF : \exists j_1 \in assignments(vars(f)) : \forall j_2 \in assignments(vars(f)) : sat(j_1, f) \geq sat(j_2, f)$.

Alternatively, when only approximate solutions are sought, an optimization problem can be specified by a singleton claim family of the form $\forall i : \forall \delta : \exists o_1 : \forall o_2 : \Psi(i, \delta, o_1, o_2)$ where $\Psi(i, \delta, o_1, o_2)$ is a logical formula that holds when o_1 is the correct output for i and o_2 is at most δ better than o_1 or when o_2 is not a correct output for i.

Promises can be added to the problem specification as a restriction on the domain of discourse. For example, in the aforementioned Bertrand's postulate example, the domain of discourse for the universally quantified variable n is $\mathbb{N} \setminus \{0, 1\}$ and the domain of discourse for the existentially quantified variable k is restricted to satisfy the following predicate $k \in \mathbb{N} \wedge n \leq k \leq 2 \cdot n$. Alternatively, promises can be directly added to claims. In this case, it is the opponent's responsibility to watch for violated promises. For example, we can formulate Bertrand's postulate as $Bertrand'() := \forall n : \neg(n \in \mathbb{N} \setminus \{0, 1\}) \vee \exists k : k \in \mathbb{N} \wedge n \leq k \leq 2 \cdot n \wedge prime(k)$. Note that promises are added to universally quantified variables using logical implications but added to existentially quantified variables using logical conjunctions.

It is worth noting that we do not restrict the structures used to interpret sentences in claims beyond the slight requirement of computability. This has the following two practical consequences:

1. It makes it convenient to express complex claims, such as claims about computer simulation models of natural phenomenas and claims about the resource consumption of algorithms, without the need to axiomatize complex computer simulation models or interpreters. For example, consider the fol-

lowing claim in the protein folding domain[3]. $\boldsymbol{HSP60}HasNativeState :=$ $\exists f \in \boldsymbol{HSP60}Foldings : \forall f_2 \in \boldsymbol{HSP60}Foldings : energy(\boldsymbol{HSP60}, f) \leq$ $energy(\boldsymbol{HSP60}, f)$ interpreted in a structure defining the constant $\boldsymbol{HSP60}$, the set $\boldsymbol{HSP60}Foldings$, the function *energy* and the predicate $\leq$. The function *energy* hides a computer simulation model for the energy of a particular protein folding.

2. It makes it possible to focus the participants on specific problems. Other problems can be scraped into the structure. For example, the aforementioned formulation of Bertrand's postulate scrapes primality testing into the structure. Alternatively, Bertrand's postulate may be formulated as : $Bertrand''() :=$ $\forall n \in \mathbb{N} \setminus \{0,1\} : \exists k \in [n, 2 \cdot n] : \forall j \in (1,k) : remainder(k, j) \geq 0$ In this formulation, the participants would be also required to search for a factor for a given number.

2.3 Organizing a Community via Semantic Games

Participants of an SG both compete and collaborate on solving the computational problems of the underlying claim. They collaborate on in the sense that the winning participant informs the losing participant about a mistake in their solutions to the computational problems of the underlying claims. In a community of participants playing a tournament of SGs, participants get a better chance to learn about more mistakes from a larger group of participants. Moreover, by playing another tourna-

[3]The folding of a protein is a 3-D structure of the protein. Proteins comprise long chains of amino acids. Certain amino acids attract, others repulse. Certain amino acids are hydrophilic and would rather be on the outside closer to water molecules, others are hydrophobic and would rather be inside away from water molecules. These forces determine the native state of the protein which is the most stable folding of the protein.

ment of SGs after the losing participants fix their mistakes, participants get another chance to fix other mistakes.

Figure 2.1 describes a workflow in which a community of participants collaborate on solving computational problems by repeatedly playing a tournament of SGs. A distinguished member of the community, the admin, is responsible for providing claims and controlling the tournaments played for each claim.

```
workflow(admin, participants){
  do{
    claim = admin provides claim;
    do{
      avatars = participants provide avatars for claim;
      (results, history) = run a tournament of SGs between
         avatars;
      notify(admin, (results, history));
      notify(participants, (results, history));
    } while(admin chooses to continue);
  }while(admin chooses to continue);
}
```

Figure 2.1: Community Collaboration Workflow

In a research community, the role of participants can be played by scholars and the role of admin can be played by a principal investigator. Claims may be chosen by the principal investigator as a part of an exploratory research for claim families with interesting properties.

In an educational community, the role of participants can be played by students and the role of admin can be played by a teacher or a professor. Claims may be chosen by the teacher according to a curriculum. The teacher may also participate as a student to point out personalized mistakes to "other" students.

In a software development community, the role of participants can be played by developers and the role of admin can be played by a manager. Claims may be chosen by the manager by adding "features".

Chapter 3

Related Work

In this chapter we describe work in three areas related to SCG. First, in Section 3.1 we describe work in the area of logical games. This work is related to our choice of semantic games to build SCG on. Then, in Section 3.3 we describe strategies for balancing the winning chances in games. This work is related to our work on balancing the winning chances in SGs. Then, in Section 3.4 we describe work in the area of tournament design. This work is related to our work of designing SCG as a tournament of SGs. Then, in Section 3.5 we describe work related to the application of SCG in crowdsourcing. Finally, in Section 3.6 we describe the origins of SCG.

3.1 Logic and Games

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 used to define truth, back-and-forth games used to compare structures, and dialogue games to express (and perhaps explain) formal proofs [31], [21], [25].

3.2 Dialogical Games vs Semantic Games

The main difference between Dialogical (or Lorenzen) games and semantic games is that dialogical games characterize validity (or logical truth) while semantic games characterize material truth (or truth in a specific model). For classical propositional and first order logics, an exact connection between "intuitionistic dialogues with hypotheses" and semantical games was shown in [36].

3.2.1 Model Checking Games

Modal logics interpreted in Kripke structures (or labeled transition systems, or labeled graphs) give rise to semantic games; truth-defining two-person games where the verifier has a winning strategy if and only if a given formula holds in a given structure [41], [21], [20].

3.2.2 Semantic Games with Retractable Moves

There are interpreted logical formulas with non-computable winning strategies for semantic games without retractable moves. Semantic games with retractable moves enable the development of recursive winning strategies [16]. Recursive strategies can undo previously taken moves and are admissible strategies for semantic games with retractable moves [16]. There are interpreted logic sentences with non-computable winning strategies yet with recursive winning strategies. the intuition behind the extra power of recursive winning strategies comes from their ability to encode a recursive adversarial search where they learn from previous mistakes [11], [9].

To illustrate recursive winning strategies, consider the formula $\exists x \forall y f(x) \leq f(y)$ where f is a parameter representing a function over the natural numbers. There is no computable function $Min(f)$ that returns the value at which f is minimum.

However, it is possible to use the following recursive winning strategy for the verifier:

1. the verifier provides 0 for x.
2. in case the falsifier is able to find some value y_1 such that $f(0) > f(y_1)$, the verifier retracts the move for x and replaces it with y_1.
3. Step 2 is repeated until the falsifier is unable to win. This will terminate because $<$ is well founded on naturals.

3.2.3 Semantic Games for Independence Friendly Logic

Independence Friendly Logic extends classical logic with a syntactic specification of quantifier dependencies. For example, the independence friendly logic sentence $(\forall x)(\exists y/\forall x)P(x,y)$ precisely expresses the situation that the existential has a constant witness regardless of the interpretation given to the universally quantified variable x. In other words, the value of y is determined without any knowledge of the value of x.

Independence Friendly Logic can express logical sentences with indeterminate truth value. We illustrate this by the following example. Consider the sentence $(\forall x)(\exists y/\forall x)|y^2-5| \geq |x^2-5|$ which can be informally described as for all x there is another value y that is closer to the square root of 5 than x, regardless of the value of x. Neither the verifier, nor the falsifier of this sentence has a winning strategy. Therefore, according to the game-theoretic semantics of Hintikka, this sentence is neither true nor false.

SGs for independence friendly logic formulas are games of imperfect information. This hiding of information makes certain SGs interesting. For example, by dropping the independence specification in the previous example, the resulting

sentence $\forall x \exists y |y^2 - 5| \geq |x^2 - 5|$ becomes trivially true with a winning strategy for the verifier defined by the Skolem function $f(x) = x$.

An alternative approach is to use encryption to hide information. For example, consider the sentence $\forall x_{enc} \exists y \forall key |y^2 - 5| \geq |decrypt(x_{enc}, key)^2 - 5|$. Although the verifier is given some information about the x chosen by the falsifier, this information is hard to utilize. Similarly, it is also hard to imagine the falsifier choosing a value for the *key* based on y.

3.3 Techniques for Balancing Winning Chances

There are games that offer asymmetric roles to players where participants in certain roles have an advantage over participants in other roles. For example, in chess there are the white and black participant roles where the white role provides the participant with the first move advantage. In soccer, each team attacks a different goal. One team can have an advantage due to wind direction for example. In SGs, participants taking the verifier role have an advantage when the underlying claim is true and participants taking the falsifier role have an advantage when the underlying claim is false.

There are *generic* approaches to restore fairness either in a single game round or across multiple game rounds. One approach is adding compensation points for the participants at a disadvantage or subtracting compensation points form participants at an advantage. An example is the Komi points added to the black participant in the game Go.

Another approach is the Pie rule in reference to a class of logical games called cut-and-choose games [21]. In the traditional cut-and-choose game one participant cuts a piece of cake into two smaller pieces; then the opponent chooses one of the pieces and eat it, leaving the other one for the cutter. This mechanism is supposed

to put pressure on the cutter to cut the cake fairly.

The pie rule can be applied to games with a demonstrated first move advantage as follows: participants are first assigned roles at random. The first participant makes a move; then the second participant gets to choose which side whether or not to swap roles with the first participant. This puts pressure on the first participant not to take advantage of the first move. This rule has been used in board games such as Hex and Mancala. It has also been applied to an extended version of go where the first move is to select the amount of Komi points to compensate the black participant. The Pie rule is not applicable to selecting a side in SGs because there are only two possible moves one that is good and one that is bad.

A third approach is to play multiple game rounds where participants alternate their roles. For example, in soccer, matches are split in halves where teams switch the goal they attack (and the team kicking off the half-match). In chess tournaments, it is often the responsibility to ensure that each participant receives, as nearly as possible, the same number of games as White and Black.

There are, also, game specific approaches to restore fairness that involve tweaking the rules of the game so that each role has a different form of advantage. For example, in soccer, one team that gets to choose the goal to attack and the other gets the kick-off. In chess, different starting configurations where the white is missing more pieces than the black were proposed [35].

3.4 Tournament Design

Tournaments turn a two-participant game into a multi-participant game comprising several two-participant games. A tournament is defined by a schedule and an evaluation algorithm. The tournament schedule specifies the number of rounds in the tournament and the two-participant games simultaneously played in each round.

The evaluation algorithm computes the standings of participants. The standings take the form of either an ordinal ranking or a cardinal rating. The output of a tournament is the standings of either all or only the top participants.

There are three major tournament formats in the literature: the round robin format, the Swiss format and the elimination (a.k.a knock out) format. Below we describe each of these major formats along with its variants.

In a round robin tournament, each participant plays once against every other participant. Which is generally considered a fair selection of each participant's opponents during the course of the tournament. Furthermore, in case ties are not possible in the underlying two-participant game, the points system is a *unique* [37], widely accepted ranking algorithm satisfying 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 which means that the relative ranking of two participants is independent of those matches in which neither is involved.

The matches of a round robin tournament of n participants are commonly modeled as the edges of the complete graph K_n. A round robin tournament schedule partitions the edge set with no two adjacent edges in the same partition [26]. Round robin tournament schedules are considered static as matches are independent of the current standings of participants. Round robin tournament schedules are chosen to optimize other psychological or logistical objectives. For example to minimize

breaks and carry over effects in the home-away patterns. An annotated bibliography of round robin tournament scheduling is given in [26].

Matches involving a "weak" player are problematic. Matches between two "weak" participants are often not interesting to spectate. Matches between "strong" and "weak" participants are also uninteresting because their results are pretty much expected. Furthermore, participants' incentives to win might be imbalanced in the later rounds of round robin tournaments. For example, winning a game in the final round might result in winning the tournament for one participant but not much for their opponent. This situation might encourage cheating.

A single elimination (a.k.a Knock Out) tournament avoids the problems of the round robin tournament format by dynamically scheduling matches only between "strong" participants. However the criteria for judging a participant to be "strong" is to win all games in the tournament. This makes elimination tournaments sensitive to minor fluctuations in participant's abilities and is consequently lowering their predictive power [38]. The predictive power of a tournament is the probability that the best participant wins the tournament. There are several variations of the elimination tournament with enhanced predictive power through biasing the tournament to the benefit of participants already known to be strong. The following paragraph summarizing these variations.

Single elimination tournaments are modeled as trees with internal nodes representing games. Participants start at the leaves and winners flow to the root. The probability of a participant winning the tournament depends on the number and the strengths of opponents it may meet on their path to the root. Some variants of the elimination tournament skew the schedule tree to shorten the paths taken by the strongest participant [45]. Other variants, such as the McIntyre System, add more paths to the root for the top participant(s). Other variants add more paths to the root for all participants such as the double elimination tournaments. In

those variants, the tournament is no longer a tree. There are also variants that dynamically seed (i.e. assign participants to nodes) the participants after each round [22], [40], [19], [45].

The Swiss tournament format avoids the problems of the round robin tournament format by dynamically scheduling matches between opponents with similar current standings. It is worth mentioning that this pairing strategy is the exact opposite of the pairing common in single elimination tournaments. In single elimination tournaments, it is desirable to *delay the confrontation* of strong participants [40] because one of the participants must be eliminated after every confrontation. Early elimination corresponds to giving the eliminated participant a low rank. There are several variations of the Swiss tournament format regarding tie breaking rules and bye rules.

3.5 Crowdsourcing and Human Computation

Crowdsourcing has become an important problem solving approach that enables us to tackle large scale problems that require human intelligence to solve. There are two main reasons that human intelligence is required to solve a problem. 1) The problem is *underspecified* such as image labeling [44], the construction of web page classifiers [23], and the creation of Wikipedia pages. Humans are needed to partially specify *what* the problem is. 2) The problem is formally specified but complex enough that we have either no known solution procedure or a rather inefficient one. Examples include, programming and discovering protein folding [15], [7]. Humans are needed either to *solve* the problem or to decide *how* to solve the problem. SCG can be used to crowdsource solutions of logically specified computational problems.

SGs provide attractive solutions to the four key challenges that crowdsourcing

systems need to address [17]:

1. What contributions can users make? SG participants are required to solve a specific set of computational problems defined by the syntax of the underlying claim.

2. How to evaluate users and their contributions? SG provides an objective basis to favor winners' solutions.

3. How to combine user contributions to solve the target problem? SGs combine the efforts of participants in the two senses described in 2.3.

4. How to recruit and retain users? SGs can be engaging to play and to spectate.

A comprehensive study of crowdsourcing is in [28]. They argue that an ideal crowd work system would offer peer-to-peer and expert feedback and encourage self-assessment. Such a system would help workers to learn, and produce better results. In SCG, participants directly communicate and may gain feedback which leads to learning. SCG provides also significant autonomy for the workers, as long as they follow the SG rules.

Kittur and Chi and Suh [27] argue that a crowdsourcing system should make the creation of believable invalid responses as effortful as completing the task in good faith. They also show that introducing verifiable questions improves the response quality significantly. The recommendation for micro-task markets is: "It is extremely important to have explicitly verifiable questions as part of the task." In SCG all requests to solve computational problems are verifiable. Indeed, in SCG it is hard to create believable invalid responses. For example, if a participant decides to intensionally take the falsifier side on a true claim, then this participant will be stuck to defending a false claim.

A socio-technical ecosystem supports straightforward integration of contributions from many participants and allows easy configuration [1]. The NSF workshop report [39] discusses socio-technical innovation through future games and virtual worlds. SGs can be easily configured through claims.

SCG provides a specific but incomplete proposal of a programming interface to work with the global brain [12]. What is currently missing is a payment mechanism for users.

SGs can be seen as a generic version of the "Beat the Machine" approach for improving the performance of machine learning systems [10] as well as other scientific discovery games, such as FoldIt [15] and EteRNA [7].

3.6 Origins

We started this line of work with the Specker Challenge Game (SCG) [30]. The goal was to create an educational game in which students can learn from each other with a minimal interaction with the teaching staff. The rules where informally described by ad-hoc rules that were called refutation protocols [1].

[1] In reference to the seminal work [34] of the famous philosopher of science, Karl Popper.

Chapter 4

The Scientific Community Game

In this chapter we address the problem of designing the Scientific Community Game (SCG) as a tournament of semantic games, guaranteeing that a perfect participant must rank at the top. The main challenge in designing SCG is that an SG can only be played by two participants *voluntarily* taking opposite sides; standard tournament formats, such as elimination, round robin and Swiss do not handle the situation where it is not possible to play a match between two participants. To adopt SGs to fit in standard tournament formats, it is inevitable to force a side on one of the participants in an SG when both participants choose the same side. Forcing a side on one of the participants creates a game with imbalanced winning chances. And consequently, there is no guarantee that a perfect participant will rank on top. There are numerous approaches to rebalance the winning chances either locally across a single match or globally across all matches in the tournament. Every approach (or combination of approaches) gives rise to an SCG design.

First, in Section 4.1 we present a number of requirements and design goals. Then, in Section 4.2 we present a number of alternative SCG designs.

4.1 Requirements and Design Goals

4.1.1 Perfect Participants Must Win

We define a perfect participant to be a participant winning all SGs when not forced. We require that a perfect participant in an SCG must be among the winners.

4.1.2 Minimizing The Advantage Given to Participants Taking the Most or Least Popular Side

Depending on the tournament format, participants taking the most popular side have a larger chance of playing against participants from the same side. Since forcing is inevitable in this case, overcompensating (respectively, undercompensating) forced players may give an undesirable advantage to participants taking the most (respectively, least) popular side. It is desirable to minimize the advantage given to participants taking the most or least popular side.

4.1.3 Maximizing The Expected Rank of The Best Participant(s)

The better the rank assigned by the tournament to the best participant, the better the tournament design. This goal is related to maximizing the predictive power [45], [38] of the tournament. Intuitively, the predictive power of a tournament is the probability that the best participant wins the tournament.

4.1.4 Minimizing The Number of Participants Ranking on Top Other Than The Best Participants

The less participants, other than the best participant ranking on top of the tournament, the better the tournament design. We refer to this quality as the discriminative power of the tournament.

4.1.5 Minimizing The Number of Semantic Games

It is also desirable to minimize the average number of SGs to be played in the context of SCG.

4.1.6 Minimizing Manipulation Chances

It is also desirable to reduce the chances of manipulating the tournament results. The key is to reduce opponent dependent behavior, such as *selectively* losing against a specific opponent.

One possibility to reduce opponent dependent behavior is to restrict participants to take the *same* side on the underlying claim in all SGs held in the context of an SCG. Another possibility is to hide the opponent identity in all SGs held in the context of an SCG. However, it is sometimes possible to infer the identity of the hidden opponent from the exchange taking place in the context of an SG. For example, if a participant is required to provide a CNF formula, they can produce formulas with a particular naming scheme to variables or with a particular structure that may be identifiable by other participants. Obfuscation can be used to further hide opponent identities. It is also possible for participants to identify their opponent through side communication outside the context of the SCG. When avatars are used, it is possible to sandbox-execute the avatars to prevent side communications.

4.2 Design Space

As we mentioned before, the naive approach of adopting a standard tournament format with a simple match setup consisting of playing a single SG with one potentially forced player might not be the best SCG design. Therefore, we present a wider SCG design space which may contain a better SCG design according to the criteria mentioned in Section 4.1. We start by presenting a number of alternative match designs in Section 4.2.1. Then, we briefly outline SCG tournament design in Section 4.2.2. Finally, in Section 4.2.3 we summarize the SCG design alternatives we presented.

4.2.1 Matches

There are two approaches to balance the winning chances in a set of SGs with a potential forced player:

- compensating forced participants with points, and
- ensuring that the forcing disadvantage is as evenly distributed as possible.

We start by presenting a number of SCG match styles aiming at distributing the forcing disadvantage as evenly as possible. Table 4.1 summarizes the match styles we presented. The first style is called the 0.5G style. In case both participants choose opposing sides, a single SG is played. In case both participants choose the same side, no games are played. However, each participant is given a point to avoid giving the least common side participants an advantage.

The second style is called the 1.0GRF style. It involves a single SG played between participants. In case participants choose the same side, one randomly chosen participant is forced to be devil's advocate. The third style is called the 1.0GRFC. It involves a single SG played between participants. In case participants

Chosen Side	0.5G	1.0GRF	1.0GRFC	2G
Opposite	1 SG	1 SG	1 SG	2 SGs
Same	No SGs each scores 1 point	1 SG Randomly force	1 SG Force least forced or randomly	2 SGs Force one at a time

Table 4.1: SCG Match Styles

choose the same side, the participant that has been forced the fewest number of times thus far is forced to be devil's advocate. If both participants were equally forced, a randomly chosen participant is forced. The forth style is called the 2G style. It involves two SGs played between participants. In case participants choose the same side, participants play devil's advocate in exactly one of the two SGs.

4.2.1.1 Scoring

Table 4.2 presents an interesting family of scoring rules for SGs with a potentially forced participant. This family of scoring rules is interesting because it satisfies the following three properties:

1. Does not encourage losing because, according to the rules, a participant can never score more by losing.

2. Does not give an advantage to participants of the most (respectively least) common side in terms of the maximum attainable score. This is because the winning participant always gets the same number of points regardless of forcing.

3. It is compensating because a non-forced winning (respectively losing) participant never scores more than a forced winning (respectively losing) participant.

Forced Participant		Compensating Score (α)	
winning	losing	winning	losing
not forced	not forced	1	0
not forced	forced	1	$0 \leq \alpha \leq 1$
forced	not forced	1	0

Table 4.2: Scoring SGs with a Potentially Forced Participant

4.2.2 Tournaments

Elimination tournament formats are not suitable for SCG because of the chances of ties in most match styles. Either the round robin or the Swiss tournament format may be used.

4.2.3 Summary

A specific SCG design in our design space can be denoted as $SCG(T \in \{RR, SW\}, M \in \{0.5G, 1GRF, 1GRFC, 2G\}, \alpha \in [0,1])$. Where *RR* denotes the round robin tournament format and *SW* denotes the Swiss tournament format. To make the design space finite, we restrict α to be in the set $\{0, 0.25, 0.5, 0.75, 1\}$.

Chapter 5

Evaluation

This chapter describes an empirical study aiming at identifying the best tournament design for SCG according to the criteria in Section 4.1. Three of the qualities mentioned in Section 4.1 require a formal characterization of the best participant(s). Such formal characterization can only be developed against a model of participants' performance in semantic games. We start in Section 5.1 by modeling participants' performance in semantic games. Then, in Section 5.2 we present our experiments. Finally, in Section 5.3 we present our results.

5.1 Modeling Participant Performance in Semantic Games

In order to evaluate tournaments, we need a formal characterization of the best participant among a set of participants. Such formal characterization can only be developed against a model of participants' performance in semantic games. The literature has two kinds of models of participants' performance; direct models and latent variable models (respectively called general models and monotonic models in [45]).

Direct models represent observable features of the participants' performance such as the pairwise winning probabilities. The main disadvantage of direct models is that sometimes there is no clear best participant and that there is no single objective approach to identify the best participant.

In latent variable models, observable features are locally independent and derived from a set of latent features. The set of latent features are chosen such that there is an obvious objective approach to identify the best participant. Examples of latent variable models in the literature include the Thurstone-Mosteller model [42] and the Bradley-Terry model [13]. Both models have participants' strength as latent features from which pairwise winning probabilities are derived.

Neither the Thurstone-Mosteller model nor the Bradley-Terry model is suitable for SGs. Both models have been designed for sports games and make the assumption that the pairwise winning probability depends on the difference of the strengths of the two participants. SGs can be perfectly played. A perfect participant is guaranteed to win an SG regardless of the strength of their opponent. Furthermore, when participants are not perfectly playing, the underlying claim may affect the SG outcome.

5.1.1 Direct Model

The direct model describes observable features of participant performance. We model the following two features of participant's performance in an SG of a specific claim:

- the probability of taking the verifier side, and
- the winning probability when taking the verifier side.

The direct model of participant performance in an SG of a specific claim is a two dimensional matrix W listing the pairwise winning probabilities of participants in the SG. The probability that participant S_i, taking the verifier side, winning against another participant S_j, taking the falsifier side, is given by the entry w_{ij} in W. The probability that participant S_i, taking the falsifier side, winning against another participant S_j, taking the verifier side, is given by $1 - w_{ji}$. The probability that participant S_i chooses to take the verifier side is given by the entry w_{ii}; in theory, a participant can choose a side by playing an SG with itself and choosing a side based on that SG outcome.

Now, we present a latent variable model for participant performance in semantic games.

5.1.2 Latent Variable Model

In our latent variable model, the performance of a participant is modeled with a single variable representing the participant strength. With this model there is an obvious definition of the best participant; the best participant is the participant with the highest strength. To give this model a meaning, we relate it to the direct model of participant performance. We do this in two steps, first we give a model for the behavior of a participant S with strength p. Then we relate participant behavior to the probability of taking the verifier side and winning as verifiers which form the direct model.

The behavior of a participant S with strength p can be described as follows:

1. It takes the correct side with probability p and takes the incorrect side with probability $1 - p$.

2. It plays the best possible strategy for the side it chooses with probability p and plays the worst possible strategy for the side it chooses with probability

$1-p$.

We define the best (respectively worst) possible strategy for a participant taking a particular side to be the deterministic strategy that maximizes (respectively minimizes) the participant's probability of winning against a randomly acting opponent. We illustrate the construction of best and worst strategy with the following example. Consider the claim $toy(0.5) := \forall x \in [0,1] : \exists y \in [0,1] : x+y > 0.5$. This is a true claim. Therefore, the correct side is the verifier side. A participant with strength p will take the verifier side with probability p and the falsifier side with probability $1-p$. The best strategy for the falsifier is to provide the smallest possible value for x which is 0. The worst strategy for the falsifier is to provide the largest possible value for x which is 1. A participant with strength p playing in the falsifier side will provide 0 for x with probability p and will provide 1 for x with probability $1-p$. The best strategy for the verifier is to provide the largest possible value for y which is 1. The worst strategy for the verifier is to provide the smallest possible value for y which is 0. A participant with strength p playing in the verifier side will provide 1 for y with probability p and will provide 0 for y with probability $1-p$.

In well understood domains, we can use the best and worst strategies for both sides to construct synthetic participants that behave according to this model. In less understood domains, we may not be able to construct the best and worst strategies yet we can assume their existence.

We now extend our model to compute the winning probabilities for a semantic game between two participants. As we mentioned before, the underlying claim may affect the game outcome. We model the effect of the underlying claim as three variables t, *wbb* (worst beats best) and *wbw* (worst beats worst). The variable t is a boolean variable representing the truth of the underlying claim. t determines the

		correct side	
		best	worst
incorrect side	best	1	*wbb*
	worst	1	*wbw*

Table 5.1: Winning Probability of Participants Taking the Correct Side

correct side to take on the underlying claim. For the claim $toy(0.5)$, the value of t is $true$. The variable *wbb* (respectively *wbw*) represent the winning probability of the participant taking the correct side and using the worst possible strategy against an opponent using the best (respectively worst) possible strategy. For deterministic strategies, the values of *wbb* and *wbw* are either 0 or 1. For example, the worst verifier strategy for $toy(0.5)$ is to provide 0 for y will be beaten by the best falsifier strategy which is to provide 0 for x. Therefore, for $toy(0.5)$ the value of *wbb* is 0. Also, the worst verifier strategy for $toy(0.5)$ beats the worst falsifier strategy for $toy(0.5)$ leading to a value of 1 for *wbw*.

Once we know the values of t, *wbb* and *wbw*, we can compute the values of direct model variables for any set of participants $S_1, \ldots, S_n$ with strengths $p_i, \ldots, p_n$. The probability of S_i taking the verifier side, denoted as w_{ii} in the direct model, is p_i if $t = true$ and $1 - p_i$ if $t = false$. The winning probability of the participant taking the correct side are shown in Table 5.1. This table demonstrates the important property of semantic games that the participant taking the correct side and using the best possible strategy shall win regardless of the strength of the opponent. However, when the participant taking the correct side uses the worst possible strategy, the probability of wining depends on the underlying claim as well as on the strength of the opponent.

According to Table 5.1, when $t = true$, the winning probability of participant

S_i, taking the verifier side, against participant S_j, taking the falsifier side, can be computed as $w_{ij} = p_i + (1 - p_i) \cdot (p_j \cdot wbb + (1 - p_j) \cdot wbw)$. When $t = false$, the winning probability of participant S_i, taking the verifier side, against participant S_j, taking the falsifier side, can be computed as $w_{ij} = 1 - (p_j + (1 - p_j) \cdot (p_i \cdot wbb + (1 - p_i) \cdot wbw))$.

Now, we proceed to describe our empirical evaluation of SCG tournament designs.

5.2 Experimental Plan

The goal of our experimental plan is to measure, for each SCG design, the following quantities:

- best participant's rank both with and without a perfect participant among participants, and
- the number of extra winners, and
- the number of SGs, and
- advantage to most popular side participants.

We measure the aforementioned quantities via two separate experiments that we describe below.

5.2.1 Experiment I

To measure the best participant's rank, the number of extra winners, and the number of SGs, we run a 1K simulations for each of the 40 tournament designs for each of the 8 different SG kinds for 2K populations of size between 10 to 30 participants

with randomly generated strengths where half of the populations have a perfect participant. Figure 5.1 shows a pseudocode of our first experiment.

```
SGKinds = the set of all 8 SGKinds;
tournaments = the set of all 40 Tournaments;
minPopulationSize = 10;
maxPopulationSize = 30;
numPopulations = 1000;
numSimulations = 1000;

for(hasPerfectParticipant ∈ {false, true}){
 for(i ∈ {1..numPopulations}){
  p = generatePopulation(minPopulationSize, maxPopulationSize,
      hasPerfectParticipant)
  best = p.bestParticipants();
  for(SGKind ∈ SGKinds){
   w = SGKind.latent2direct(p);
   for(tournament ∈ tournaments){
    for(j ∈ {1,...,numSimulations}){
      Result result = simulate(tournament, w);
      yieldBestsRank (tournament, hasPerfectParticipant,
         result.getRankOf(best));
      yieldNumExtraWinners (tournament, hasPerfectParticipant,
          result.getNumExtraWinners(best));
      yieldNumSGs (tournament, result.getNumSGs());
    }
   }
  }
 }
}
```

Figure 5.1: Experiment I

5.2.2 Experiment II

To measure the winning chances of participants taking the most popular side, we use a winning probability matrix that gives each participant a 0.5 probability of winning regardless of the side. Diagonal entries are set to either a 1 or a 0 making participants always take either the verifier or the falsifier side during the simulation. Without loss of generality we set more entries to 1.

We run 1K simulations for each tournament design with the aforementioned winning probability matrix for a population of a 100 participants for a fraction of verifiers ranging from 51% to 99%. We measure the fraction of verifiers among the winners. A 75% winning chance for verifiers matches the fraction of verifiers accross all populations. Therefore, a 75% winning chance for verifiers means that verifiers are not given any additional advantage that can be attributed to taking the most popular side. Figure 5.2 shows a pseudocode of our second experiment.

```
tournaments = the set of all 40 Tournaments;
populationSize = 100;
numSimulations = 1000;

for(tournament ∈ tournaments){
 numWinners = 0;
 numWinningVerifiers = 0;
 for(numVerifiers ∈ {51..99}){
  w = populationSize by populationSize matrix with
      numVerifiers 1's on the diagonal. The rest of the
      diagonal has 0's. The rest of the matrix has 0.5's.
  for(i ∈ {1..numSimulations}){
   Result result = tournament.simulate(w);
   for(winner ∈ result.getWinners()){
    numWinners++;
    if(winner.isVerifier()) numWinningVerifiers++;
   }
  }
 yieldWinningChances(tournament, numWinningVerifiers/
     numWinners);
}
```

Figure 5.2: Experiment II

5.3 Results and Discussion

The results of our experiments are summarized in Tables 5.2, 5.3. For each Tournament design, we report:

- the average and the standard deviation of best participant's rank both with and without a perfect participant among participants, and
- the average and the standard deviation of the number of extra winners both with and without a perfect participant among participants, and
- the average number of SGs, and
- the average winning chances of participants taking the most popular side when both verifiers and falsifiers have an equal chance of winning an SG.

We made the following observations about the results:

- There are only 16 designs where a perfect participant is guaranteed to rank at the top. In all of these 16 designs the forced player is fully compensated with 1 point. There is a simple explanation to this observation; full compensation, makes a perfect participant scores the maximum attainable score in an SCG because a perfect participant, by definition, will win when not forced, and consequently score a point. Also, a perfect participant will also score the compensation point when forced. Since all participants have the same maximum attainable score, a perfect participant is guaranteed to rank at the top.
- In the absence of a perfect participant, fully compensating the forced player with 1 point lowers the average rank of the best participant. There is no similar conclusion that can be drawn for other amounts of compensations.
- Whether or not there is a perfect participant, fully compensating the forced player with 1 point significantly increases the average number of participants ranking at the top. There is no similar conclusion that can be drawn for other amounts of compensations.

Tournament	Without a Perfect Player				With a Perfect Player				# of SGs	Popular Winning Chance
	Best's Rank		Extra Winners		Best's Rank		Extra Winners			
	Avg	SD	Avg	SD	Avg	SD	Avg	SD		
SCG(RR, 0.5G, 0.0)	1.827	2.737	4.048	3.93	1	0.003	4.454	3.881	105.949	0.99
SCG(RR, 0.5G, 0.25)	1.826	2.735	4.047	3.93	1	0.002	4.454	3.881	105.949	0.99
SCG(RR, 0.5G, 0.5)	1.828	2.736	4.047	3.931	1	0	4.454	3.882	105.948	0.99
SCG(RR, 0.5G, 0.75)	1.828	2.735	4.047	3.93	1	0.001	4.454	3.882	105.946	0.99
SCG(RR, 0.5G, 1.0)	1.827	2.735	4.048	3.93	1	0.002	4.455	3.882	105.953	0.99
SCG(RR, 1GRF, 0.0)	3.407	3.379	1.082	0.835	3.008	2.669	1.067	0.837	211.849	0.75
SCG(RR, 1GRF, 0.25)	3.45	3.428	0.918	0.742	2.956	2.614	0.918	0.758	211.849	0.87
SCG(RR, 1GRF, 0.5)	3.232	3.422	0.964	0.792	2.697	2.5	0.95	0.805	211.849	0.92
SCG(RR, 1GRF, 0.75)	3.191	3.473	0.892	0.768	2.552	2.443	0.878	0.79	211.849	0.94
SCG(RR, 1GRF, 1.0)	1.885	2.935	3.452	3.99	1	0.005	3.766	3.966	211.849	0.96
SCG(RR, 1GRFC, 0.0)	3.241	3.289	1.145	0.925	2.934	2.806	1.093	0.954	211.849	0.75
SCG(RR, 1GRFC, 0.25)	3.307	3.358	0.974	0.842	2.889	2.751	0.948	0.879	211.849	0.86
SCG(RR, 1GRFC, 0.5)	3.125	3.37	1.019	0.884	2.644	2.629	0.978	0.916	211.849	0.92
SCG(RR, 1GRFC, 0.75)	3.11	3.428	0.951	0.864	2.518	2.567	0.916	0.901	211.849	0.94
SCG(RR, 1GRFC, 1.0)	1.88	2.926	3.447	3.991	1	0.003	3.759	3.969	211.849	0.96
SCG(RR, 2G, 0.0)	1.919	2.45	2.909	4.163	1.33	0.712	3.013	4.226	423.699	0.75
SCG(RR, 2G, 0.25)	1.976	2.575	2.776	4.212	1.322	0.698	2.877	4.275	423.699	0.89
SCG(RR, 2G, 0.5)	1.917	2.614	2.798	4.208	1.243	0.574	2.891	4.276	423.699	0.94
SCG(RR, 2G, 0.75)	1.941	2.698	2.757	4.226	1.217	0.524	2.848	4.294	423.699	0.96
SCG(RR, 2G, 1.0)	1.799	2.747	2.961	4.155	1	0.007	3.141	4.188	423.699	0.97

Table 5.2: Experimental Results: Part I

Tournament	Without Perfect Players				With Perfect Players				# of SGs	Popular Winning Chance
	Best's Rank		Extra Winners		Best's Rank		Extra Winners			
	Avg	SD	Avg	SD	Avg	SD	Avg	SD		
SCG(SW, 0.5G, 0.0)	1.66	2.795	10.768	4.416	1	0.004	10.763	4.321	11.876	0.91
SCG(SW, 0.5G, 0.25)	1.659	2.794	10.769	4.416	1	0.002	10.762	4.321	11.876	0.91
SCG(SW, 0.5G, 0.5)	1.661	2.797	10.769	4.417	1	0	10.762	4.321	11.876	0.91
SCG(SW, 0.5G, 0.75)	1.66	2.796	10.771	4.415	1	0.003	10.763	4.321	11.873	0.91
SCG(SW, 0.5G, 1.0)	1.66	2.793	10.768	4.415	1	0.002	10.764	4.322	11.876	0.91
SCG(SW, 1GRF, 0.0)	3.798	3.507	1.28	1.23	3.457	3.022	1.248	1.213	47.071	0.75
SCG(SW, 1GRF, 0.25)	4.508	4.263	1.222	1.004	4.085	3.848	1.225	1.029	47.071	0.80
SCG(SW, 1GRF, 0.5)	3.818	3.431	1.203	0.963	3.333	2.701	1.204	0.985	47.071	0.80
SCG(SW, 1GRF, 0.75)	3.866	3.458	1.193	0.945	3.286	2.589	1.193	0.966	47.071	0.80
SCG(SW, 1GRF, 1.0)	1.848	2.889	6.325	3.503	1	0	6.687	3.417	47.071	0.90
SCG(SW, 1GRFC, 0.0)	3.691	3.437	1.554	1.465	3.327	2.922	1.545	1.467	47.071	0.75
SCG(SW, 1GRFC, 0.25)	4.406	4.252	1.616	1.308	3.965	3.815	1.643	1.334	47.071	0.81
SCG(SW, 1GRFC, 0.5)	3.748	3.425	1.539	1.186	3.246	2.67	1.559	1.207	47.071	0.81
SCG(SW, 1GRFC, 0.75)	3.818	3.47	1.503	1.142	3.229	2.581	1.523	1.163	47.071	0.81
SCG(SW, 1GRFC, 1.0)	1.842	2.885	6.29	3.507	1	0.004	6.651	3.423	47.071	0.91
SCG(SW, 2G, 0.0)	3.184	3.055	2.364	2.404	2.825	2.473	2.451	2.447	94.143	0.75
SCG(SW, 2G, 0.25)	3.319	3.22	2.169	2.491	2.792	2.401	2.239	2.485	94.143	0.79
SCG(SW, 2G, 0.5)	2.974	3.062	2.297	2.506	2.425	2.087	2.364	2.505	94.143	0.83
SCG(SW, 2G, 0.75)	3.006	3.165	2.241	2.502	2.344	2.019	2.313	2.503	94.143	0.87
SCG(SW, 2G, 1.0)	1.862	2.752	4.859	3.778	1	0.003	5.285	3.706	94.143	0.91

Table 5.3: Experimental Results: Part II

- When any compensation at all is given to forced players, the wining chances of participants of the popular side dramatically increase (assuming that participants of both sides have the same winning chance in a single SG). Fully compensating the forced player with 1 point brings the the winning chances of participants of the popular side close to 100%.

- By Focusing on the 16 designs where a perfect participant is guaranteed to rank at the top and ignoring that the advantage given to participants taking the most popular side we can make the following observations:

- 1. There is no significant differences regarding the average rank of the best player, and

  2. the design SCG(RR, 2G, 1.0) minimizes the number of extra winners but requires significantly more SGs than others.

Chapter 6

The Formal Science Wikipedia

[TBD] Describe the syntax for claims and give examples.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This dissertation describes our research on developing the scientific community game as an alternative approach to organizing scientific research competitions for computational problems. SCG is essentially a tournament of semantic games, guaranteeing that a perfect participant must rank at the top.

The main challenges we addressed were:

- In developing SCG: standard tournament formats, such as elimination, round robin, and Swiss, cannot be used with SGs because SGs can only be played between two participants *voluntarily* taking opposite sides. To adopt SGs to fit in standard tournament formats, it is inevitable to force a side on one of the participants in an SG when both participants choose the same side. Forcing a side on a randomly chosen participant creates a game with imbalanced winning chances. Consequently, there is no guarantee that a perfect participant will rank on top.

- In evaluating SCG: a formal characterization of the best participant in a given

set of participants is needed. This can be done in two different ways, using a reference tournament or using a model of the performance of a participant of a given strength in an SG. Unfortunately, there is no clear objective way to decide on a reference tournament. Furthermore, existing models for participants' performance based on their strength such as the Thurstone-Mosteller model [42] and the Bradley-Terry model [13] where developed for sports games and their underlying assumptions are not preserved by SGs.

In this dissertation we described:

- how SCG can be used to organize scientific research competitions, and
- a number of additional design goals for SCG, and
- a number of SCG designs based on adopting SGs to two standard tournament formats, and
- a model for participant's performance in SG, and
- an empirical evaluation of SCG designs we presented, and
- a description of the formal science wikipedia, our web based SCG implementation.

Based on our empirical evaluation of the SCG designs we presented, we conclude that when standard Swiss or round robin tournament formats are used, it is inevitable to fully compensate forced participants with a point in order to guarantee that a perfect participant will rank on top. On the other hand, none of the SCG designs guaranteeing that a perfect participant will rank at the top, among the SCG design alternatives we considered, reasonably satisfy the design goals we presented. Further research into non-standard tournament formats is necessary.

7.2 Future Work

As a future work, we want to:

1. Consider other SCG designs with non-standard tournament formats. Examples of non-standard formats we want to consider include:

    a) Separately rank participants taking the verifier side and participants taking the falsifier side, then somehow combine both rankings. The goal is to lower the advantage given to the most popular side.

    b) Develop a Hybrid version between the Swiss and the Round Robin tournament. The goal is to lower the average number and at the same time minimizing the number of extra winners.

2. Utilize relations between claims in the formal science wikipedia to effectively merge communities organized around claims.

REFERENCES

[1] Center on Architecting Socio-Technical Ecosystems. Website. http://www.coaste.org/.

[2] Kaggle. Website. http://www.kaggle.com/.

[3] Netflix Prize. Website. http://www.netflixprize.com/.

[4] Piazza. Website. http://www.piazza.com.

[5] Project Euler. Website. http://projecteuler.net/.

[6] The international SAT Competitions. Website. http://www.satcompetition.org/.

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

[8] Ahmed Abdelmeged and Karl Lieberherr. The Formal Science Wikipedia. Website, 2013. http://www.fswikipedia.net/.

[9] Federico Aschieri. *Learning, Realizability and Games in Classical Arithmetic*. PhD thesis, Universit'a degli Studi di Torino Dipartimento di Informatica and Queen Mary, University of London School of Electronic Engineering and Computer Science, 2011.

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

[11] Stefano Berardi. Semantics for intuitionistic arithmetic based on tarski games with retractable moves. In *Proceedings of the 8th international conference*

on Typed lambda calculi and applications, TLCA'07, pages 23–38, Berlin, Heidelberg, 2007. Springer-Verlag.

[12] Abraham Bernstein, Mark Klein, and Thomas W. Malone. Programming the global brain. *Commun. ACM*, 55(5):41–43, May 2012.

[13] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[14] Dan Cohen. Learning to lose to learn: A funny thing about arguments: at. TEDxColbyCollege, 2013. http://www.youtube.com/watch?v=-5-QZugvy0g.

[15] Seth Cooper, Adrien Treuille, Janos Barbero, Andrew Leaver-Fay, Kathleen Tuite, Firas Khatib, Alex Cho Snyder, Michael Beenen, David Salesin, David Baker, and Zoran Popović. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 40–47, New York, NY, USA, 2010. ACM.

[16] Thierry Coquand. A semantics of evidence for classical arithmetic. *The Journal of Symbolic Logic*, 60(1):pp. 325–337, 1995.

[17] Anhai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, April 2011.

[18] Richard Galant. What if students learn faster without teachers? The New York Times, 2001. http://www.cnn.com/2013/02/27/opinion/ted-prize-students-teach-themselves.

[19] Mark E. Glickman. Bayesian locally optimal design of knockout tournaments. *Journal of Statistical Planning and Inference*, 138(7):2117 – 2127, 2008.

[20] Erich Grädel. Model checking games, 2002.

[21] Wilfrid Hodges. Logic and games. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2009 edition, 2009.

[22] F. K. Hwang. New concepts in seeding knockout tournaments. *The American Mathematical Monthly*, 89(4):pp. 235–239, 1982.

[23] P. Ipeirotis, F. Provost, V. Sheng, and J. Wang. Repeated labeling using multiple noisy labelers. *This work was supported by the National Science Foundation under GrantNo. IIS-0643846, by an NSERC P, Vol*, 2010.

[24] George Johnson. The world: In silica fertilization; all science is computer science. The New York Times, 2001. http://www.nytimes.com/2001/03/25/weekinreview/the-world-in-silica-fertilization-all-science-is-computer-science.html.

[25] Laurent Keiff. Dialogical logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2011 edition, 2011.

[26] Graham Kendall, Sigrid Knust, Celso C. Ribeiro, and Sebastin Urrutia. Scheduling in sports: An annotated bibliography. *Computers Operations Research*, 37(1):1 – 19, 2010.

[27] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 453–456, New York, NY, USA, 2008. ACM.

[28] Aniket Kittur, Jeffrey V. Nickerson, Michael Bernstein, Elizabeth Gerber, Aaron Shaw, John Zimmerman, Matt Lease, and John Horton. The future of crowd work. In *Proceedings of the 2013 conference on Computer supported cooperative work*, CSCW '13, pages 1301–1318, New York, NY, USA, 2013. ACM.

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

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

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

[32] Jordi Petit, Omer Giménez, and Salvador Roura. Jutge.org: an educational programming judge. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, pages 445–450, New York, NY, USA, 2012. ACM.

[33] Ahti Pietarinen. Games as formal tools vs. games as explanations. Technical report, 2000.

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

[35] Thomas H. Quinn. Level the playing field: Nullifying first-move advantage in chess. http://cargocollective.com/tomquinn/Level-the-Playing-Field-Nullifying-First-Move-Advantage-in-Chess, may 2011.

[36] Shahid Rahman and Tero Tulenheimo. From games to dialogues and back. In Ondrej Majer, Ahti-Veikko Pietarinen, and Tero Tulenheimo, editors, *Games: Unifying Logic, Language, and Philosophy*, volume 15 of *Logic, Epistemology, and the Unity of Science*, pages 153–208. Springer Netherlands, 2009.

[37] Ariel Rubinstein. Ranking the participants in a tournament. *SIAM Journal on Applied Mathematics*, 38(1):pp. 108–111, 1980.

[38] Dmitry Ryvkin and Andreas Ortmann. Three prominent tournament formats: Predictive power and costs. Cerge-ei working papers, The Center for Economic Research and Graduate Education - Economic Institute, Prague, 2006.

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

[40] Allen J. Schwenk. What is the correct way to seed a knockout tournament? *The American Mathematical Monthly*, 107(2):pp. 140–150, 2000.

[41] C. Stirling. *Modal and Temporal Properties of Processes*. Texts in Computer Science. Springer, 2001.

[42] Louis L Thurstone. A law of comparative judgment. *Psychological review*, 34(4):273, 1927.

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

[44] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM.

[45] Thuc Duy Vu. *Knockout Tournament Design: A Computational Approach*. PhD thesis, Stanford University, Department of Computer Science, 2010.