The Scientific Community Game: A Lens to Focus the Global Brain

Karl Lieberherr
Northeastern University
CCIS
Boston
lieber@ccs.neu.edu

Ahmed Abdelmeged
Northeastern University
CCIS
Boston
mohsen@ccs.neu.com

ABSTRACT

We define the Scientific Community Game (SCG, formerly called the Specker Challenge Game) and we show basic game properties which are key to making the game interesting. An example illustrates the concepts. SCG, the first generic model of the Popperian Scientific Method on the web, models scientific communities and has broad applications in teaching and research organization.

SCG is designed to be both educational for scholars, and to solve problems that we don't know how to solve yet. SCG provides a systematic framework to develop and disseminate the world's constructive claims in formal scientific domains. The development of claims is both collaborative and self-evaluating, and SCG is an effective lens to focus the global brain on solving a specific problem.

A key contribution of the paper is the design of SCG, including a CSP-model of payoff function design. The CSP-model allows us to state and prove key properties of the game across all its instantiations.

12/19/2012 v4

Keywords

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

1. INTRODUCTION

Popper, one of the most prominent philosophers of science of the previous century, promoted in "Conjectures and Refutations" the idea that each hypothesis should have a description how it could be refuted, and that scientific knowledge grows through the introduction of refutable hypothesis followed by intensive testing of their correctness.

We designed the Scientific Community Game (SCG) to encourage scholars to put as much effort and intelligence as possible into proposing falsifiable claims as well as into disputing those claims. The gamification of science that we propose has several **benefits**: (1) We make scientific activity available to a wider audience. Trying to refute a claim is a simple activity. (2) We get psychological advantages: playing the game is fun and leads to the discovery of what is known within the current group of players. (3) We make it easy to expose claims to refutation attempts, which leads to better science in the long run [**?**]. (4) The lab definition mechanism of the SCG makes it easy to focus a group of people on a specific problem. (5) Popper's ideas become applicable to claims in any formal science not only to complex scientific theories.

To be submitted for review to the Serious Games track of the conference Foundations of Digital Games, 2013, Crete.

1.1 SCG in a Nutshell

In the SCG, a lab consists of a set of refutable claims [**?**]. Scholars take on the roles proponents and opponents of the claims in the lab. Opponents attempt to refute claims against proponents. A refutation attempt initiates an orderly dialog between the proponent and opponent, which induces collaborative behavior. In the SCG, successfully refuting a claim against the claim's proponent does not mean that the claim is false. It only means that the proponent might not have the skills to defend the claim. The labs are self-evaluating in that there is no need of a third party to evaluate the contributions of the scholars: the proponent and opponent evaluate each other fairly.

There are two classes of SCG users, lab designers and scholars (players).

- Lab designers have a problem they want to focus scholars on solving through specializing SCG. The role of lab designer can be played by a researcher who needs a specific problem solved or an educator who wants her students to practice what they were taught. Lab designers define a few sets and functions that define the problem to be solved. As return on their investment they get a lab where scholars are invited to participate to solve the problem. Defining a lab can be viewd as writing a program for the global brain [**?**].

- Scholars inhabit a lab, either because they are enticed by an educator or because they have good skills to solve the problem defined by the lab. A scholar has the opportunity to influence the quality of the knowledge base and the quality of the lab procedures. Scholars are evaluated in the lab based on the quality of their contributions compared to the contributions of their peers.

  The role of scholar can be played by a human or an avatar (software). The avatar variant works only for labs that are sufficiently well understood.

1.2 SCG Applications

SCG has several **applications**, including:

1. problem solving and research in formal science. Funding agencies, such as NSF, define, in collaboration with interested researchers, labs that define the problem to be solved.

Through playing the game, NSF builds a knowledge base of refutable claims and refutation attempts. Furthermore, the self-evaluating nature of SCG will fairly evaluate the contributions of scholars and the collaborative nature will lead to productive team work. Newcomers can contribute by participating in a long-running lab (dozens of years).

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

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

1.3 Contributions

The main contribution of this paper is a definition of SCG, a generic game for the Popperian Scientific Method [**?**]. A secondary contribution of the paper is a formalization of refutations based on earlier work in logic (predicate logic and its generalizations). We also show that the SCG has several **desirable properties** (see section **??**).We have developed an implementation of the SCG which is available on SourceForge [**?**]. The SCG definition in this paper presents an abstraction of our implementation.

1.4 Organization

TBD.

2. BINARY GAME DEFINITION

We first define the binary SCG which is the building block for competitions. The game is played between two scholars (players), Alice and Bob, disputing the correctness and optimality of claims about a particular domain. We define the binary SCG through its extensive-form (or tree-form) representation which is a common representation in game theory [**?**]. We start by giving some auxiliary definitions for *Domain*, *Refutation Protocol*, *Lab*, *Claim* and *Refutation Game* that we eventually use for defining the Binary SCG.

2.1 Domain

A *Domain* consists of:

1. a set *Instance* of (problem) instances,

2. a set *Solution* of (problem) solutions,

3. a predicate *valid(i : Instance, s : Solution) : Boolean* that holds when the solution *s* is valid for the instance *i*, and

4. a function *quality(i : Instance, s : Solution) : Real* that returns a real number representing the quality of the solution *s* to the instance *i*.

When designing a domain the *valid* function should be as lax as possible such that valid solutions for any problem can be found with minimal to no intelligence. The reason is that failing to provide a *valid* solution is used as an indicator that a player (which can be an avatar) is out of order and that it should be kicked out of the lab immediately. It should also be kept in mind when designing the *quality* function that the game interprets higher quality as being better.

Examples of domains include, the domain of CNF Satisfiability where *Instance* is the set of CNF formulae, *Solution* is the set of variable assignments to booleans. A solution *s* is valid for an instance *i* if and only if *s* provides a boolean value for all variables mentioned in *i*. A potential definition of *quality(i,s)* is the fraction of clauses in *i* satisfied by *s*.

When there are multiple valid solutions for a given problem, the $quality$ function is used to discriminate between them. For example, for the domain where $Instance$ is the set of sequence alignment and $Solution$ is the set of all sequence alignment algorithms such as Smith-Waterman and BLAST,...etc, we could be concerned with full sequence alignment in which case, Smith-Waterman might have a higher quality. If we are interested in trading some accuracy for speed, BLAST might be of a Higher quality.

2.2 Refutation Protocol

A refutation protocol is a sequence of actions to be taken by two players, a proponent and an opponent of a particular claim, in order to resolve a dispute about whether the claim holds or not. The actions can be either to provide an instance or to provide a solution to a particular instance. Protocols can be defined by the following grammar:

```
ProtocolSpec =  <steps> CommaList(Step).
Step = <actor> Actor <action> Action.
Actor = Proponent | Opponent.
Proponent = "P".
Opponent = "O".
Action = ProvideAction | SolveAction.
ProvideAction = "I" "[" <instanceId> int "]".
SolveAction = "S" "[" <solutionId> int "]"
  "of" <instance> ProvideAction.
```

An example of a protocol is $P : I[0], O : S[1]\ of I[0]$ in which the proponent provides an instance named $I[0]$ and the opponent provides a solution named $S[1]$ for the instance $I[0]$.

2.3 Lab and Claim

Labs play a central role in SCG. SCG games are played in the context of a particular lab. A lab focuses the scientific discourse of games played in it through defining a family of claims about a particular domain. The claims in a lab have the same structure and only differ in the claim parameter values. Labs require a great care to define and effectively resemble a declarative specification of a (human) computation [**?**] that separates true from false claims in the lab and also finds the optimal claims in optimization labs.

A *Lab* consists of:

1. a $Domain : d$ that gives a context for claims defined in the lab;

2. a refutation protocol $proto : Protocol$, used to resolve disputes regarding the correctness of claims;

3. a claim structure $Claim$ consisting of:

   (a) a number of claim parameters;

   (b) an instance set predicate $isp(i : d.Instance) : Boolean$ defining a family of instance sets. Each set in the family is a subset of *d.Instance* and is chosen by claim parameter values. The intuition is that a claim c asserts a property about the set $\forall i \in d.Instance : c.isp(i)$.

(c) a refutation predicate $p(I : d.Instance[], S : d.Solution[]) : Boolean$ defines the property that the claim asserts about the family of sets defined by isp. The protocol together with the refutation predicate are well-formed, if every ProvideAction is used by at least one later SolveAction or the refutation predicate and every SolveAction refers (through the step number) to a ProvideAction. The output of every SolveAction must be used in the refutation predicate.

(d) a predicate $stronger(c_2 : Claim) : Boolean$ defining a partial order on claims in the lab.

(e) a function $distance(c_2 : Claim) : Real$ defining the distance between two claims in the lab provided that the one of the claims is stronger than the other.

Consider a lab with the following protocol instance $P : I[0], O : S[1]$ of $I[0]$. A claim c in this lab intuitively means: "I, the proponent, can give an instance $I[0]$ where $c.isp(I[0])$ holds, such that for any solution *S[1]* of *I[0]* given by you, the opponent, where *valid(I[0],S[1])* holds, the refutation predicate $p(I, S)$ holds.". This claim can also be expressed as a mathematical statement about the underlying domain as: "There exists an instance *I[0]* where $c.isp(I[0])$ holds, such that for all solutions *S[1]* of *I[0]* where *valid(I[0],S[1])* holds, the refutation predicate $p(I, S)$ holds.". In order to support this claim, the proponent has to deliver $I[0]$ and opponent has to deliver *S[1]* in a resource constrained environment.

If we append the following protocol step to the end of the above protocol $P : S[2]$ of $I[0]$ and define the refutation predicate to be: $p(I, S) = (Lab.d.quality(S[2], I[0]) \geq Lab.d.quality(S[1], I[0]))$, we get a very different kind of claim. It says that proponent is at least as good as opponent in solving instances in the given lab. Which is a statement about the performance of players rather than about the underlying domain.

2.3.1 Classification of Claims and Labs

We define a claim to be **true** if it has a support strategy for the proponent. This means that, for true claims, the proponent can always avoid refutations. We define a claim to be **false** if it has a refutation strategy for the opponent, i.e., the opponent can always succeed in refuting. We define a claim to be **indeterminate** if it has neither a defense nor a refutation strategy.

We classify labs into **optimization**, **standard**,and **bivalent** labs. Optimization labs have no restrictions. Standard labs are where claims are not comparable for strength (i.e. the $stronger$ predicate is defined to be $false$). Bivalent labs are a special case of standard labs with no indeterminate claims.

2.4 Refutation Game

To attempt a refutation of a given claim c, both the proponent pro and the opponent opp of the claim enter into a refutation game $refute(c : Claim, pro : Scholar, opp : Scholar)$.

The refutation game consists of an exchange of instances and solutions as defined by the protocol $c.Lab.proto$ where the proponent P and opponent O in the protocol are bound to the scholars pro and opp respectively. The exchanged instances must be in the set $\forall i \in c.d.Instance : c.isp(i)$. A solution $S[n]$ for instance $I[m]$ must be valid for that instance. i.e. $c.d.valid(I[n], S[m])$ must hold. The refutation predicate is used to decide the winner of the refutation game. If the $c.p(I, S)$ holds, the proponent has successfully supported the claim and wins. Otherwise, the opponent has successfully refuted the claim and wins. I, S refer to the instances and solutions exchanged during the refutation game. As a general rule, solutions are all kept secret until protocol evaluation. When a scholar provides a solution she does not know about solutions provided by the other scholar.

2.5 Binary SCG

The first move is performed by the scholar taking the role of the proponent, and it is to propose a claim c from the claims in the lab. By doing so, the proponent asserts that c is a true claim that there is no other claim in the lab that is stronger. In Figure 1, the first move is denoted by a thick dashed blue line to indicate that it is a multi-edge, an edge for each possible claim c.

The second move is performed by the opponent, and it is to decide whether to:

1. dispute the correctness of c, denoted *disputeD(c)*, or to
2. dispute the optimality of c. In this case, the opponent must provide a stronger claim c'. This decision is denoted *strengthenD(c, c')*, or to
3. agree with c, denoted *agreeD(c)*.

Based on the second move, the proponent and the opponent enter into a different refutation game, which is denoted by a thick solid red line to indicate that it is a path. If the second move is *disputeD(c)*, the refutation game is $refute(C,$ proponent , opponent). If the second move is *strengthenD(c, c')*, the refutation game is $refute(C'$, opponent , proponent). If the second move is *agreeD(c)*, the refutation game is $refute(C,$ opponent , proponent). Even though, there is no dispute, the two players engage in a refutation game to test whether the opponent is indeed able to support c or it is only trying to avoid entering into a dispute with the proponent over c.

Scholars must make their moves within a given resource (time, space, ...etc) limit. The winner of the binary SCG, is the scholar that achieve the highest payoff. Payoff is disscussed in section 2.5.1.

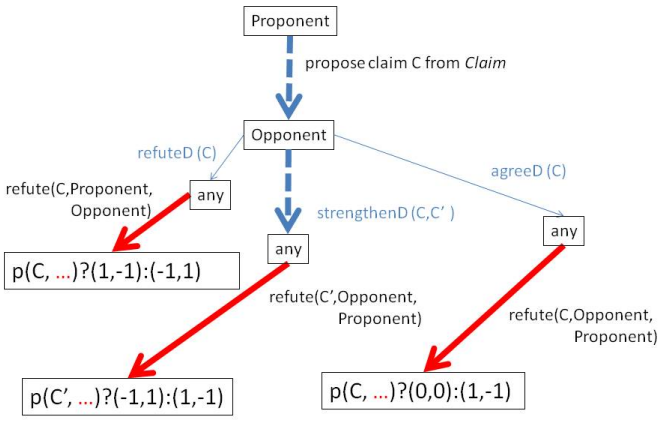

Figure 1: SCG Binary Game Tree.

2.5.1 Payoff function

Payoff is determined by: (1) the decision of the opponent in the second move. (2) the value of a refutation predicate which has as parameters the values collected along the current path back to

the root. Including either the proposed claim c or the strengthened claim c', as well as instances I and solutions S provided during the refutation game.

We use the following notation to describe the payoff function: $c.p(I,S)?(\mathrm{Pt},\mathrm{Ot}),(\mathrm{Pf},\mathrm{Of})$ If $c.p(I,S)$ is true, Pt is the payoff for the proponent and Ot is the payoff for the opponent. If $c.p(I,S)$ is false, Pf is the payoff for the proponent and Of is the payoff for the opponent.

1. If the opponent decision is *disputeD(c)*, the payoff is $c.p(I, S)\ ?\ (1,-1) : (-1,1)$. The rationale is that if the predicate holds, the proponent has successfully supported c and gets a point for that. The proponent failed to refute c and loses a point. If the predicate is false, the opponent has failed to support c and loses a point. The opponent has successfully refuted the c and gets a point.

2. If the opponent decision is *strengthenD(c, c')*, the payoff is $c'.p(I,\ S)\ ?\ (-d,\ d) : (1,\ -1)$. where $d = c.distance(c,\ c')$. The rationale is that if the predicate holds, the opponent gets rewarded because she successfully supported her proposed stronger claim c'. The payoff is proportional to the amount of strengthening as specified by the distance function to encourage more strengthening. If the predicate does not hold, the proponent has successfully refuted the stronger claim and gets rewarded with a point. The strengthening was not successful.

3. If the opponent decision is *agreeD(c)*, the payoff is $c.p(I,\ S)?\ (0,\ 0)\ :\ (1,\ -1)$. The rationale is that if the predicate holds, the opponent has successfully supported the claim and nobody gets a point because no dispute has taken place. Agreement is successful. If the predicate does not hold, the opponent has failed to support the claim and loses a point. Agreement is not successful.

2.6 Example

Fig. 2 (top left) provides a summary of the *Domain* structure. Fig. 2 (top right) provides an example of a *Domain*, the Gale-Shapely worst case domain. In this domain, *Instance* is the set of natural numbers, *Solution* is the set of preferences which is the input to the Gale-Shapely matching algorithm. A solution *s* is valid for an instance *i* if and only if *s* is syntactically well formed and it gives preferences for *i* men and women. The quality of a solution *s* to an instance *s* is the number of iterations of the while loop in the Gale-Shapely matching algorithm when supplied the preferences *s* as input.

Fig. 2 (middle left) provides a summary of the *Lab* structure and provides an example *Lab* (middle right), the Gale-Shapley worst case lab. *Claim*s in this lab have two parameters n and q. n is the number of men and women. q is the number of iterations of the Gale-Shapely matching algorithm that can be achieved by giving a preference for n men and women. The instance set predicate of the claim selects only one instance, namely the instance where the number of men and women is the same as the claim parameter n. The refutation protocol is that the opponent should provide an instance $I[0]$ satisfying the instance set predicate of the claim (there is only a singleton instance) and the proponent should provide a preference $S[1]$ for $I[0]$. The refutation predicate holds if $S[1]$ causes the Gale-Shapley matching algorithm to iterate at least q times. The $stronger$ predicate holds if the current claim has a heigher value for the q parameter. The distance function $distance$ is defined as the difference between the q parameter of the current and the given claim. Several other examples in the style of Fig. 2 are in the supplementary materials [**?**].

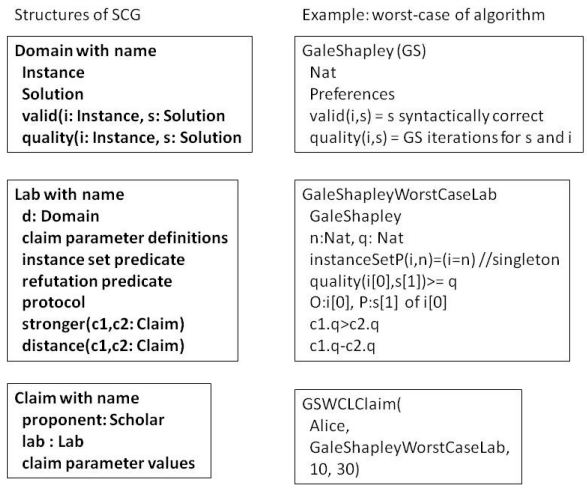

Figure 2: SCG Structure.

3. BEYOND BINARY SCG

3.1 Matches and Tournaments

To alleviate potential first-move-advantage in binary SCG, we propose that players engage in matches rather than binary games. A match consists of a number of rounds of binary SCG where players switch the roles of opponent and proponent. Tournaments (e.g. round-robin, knock-out, Swiss-style) enable more than two players to engage in a single competition. Round-robin tournaments are more suitable for avatars playing SCG because they involve $n.(n-1)$ matches where n is the number of players. Swiss-style tournaments are more suitable for humans playing SCG because it involves fewer number of games.

3.2 Reputation Score

In real scientific communities, scientists build up their reputation based on their work and the breakthroughs they achieve. Likewise, in SCG, we propose to compute reputation scores, achieved in a particular lab, for scholars based on their performance in competitions held in that lab as well as on their breakthroughs.

3.2.1 Breakthroughs

Being the first in a scientific community is crucial for the reputation of a scholar. Therefore, the SCG as a faithful model of a scientific community, also deals with being the first.

The breakthrough metric is computed after a competition or a series of competitions as a retrospective. For each important event, a time stamp is stored. Important events are: refutation, support and strengthening. We assume that we have the set of claims believed to be true (*BelievedTrue*) or false (*BelievedFalse*) or optimal (*BelievedOptimal*).

For each claim in *BelievedTrue* or *BelievedOptimal*, we find the first scholar who proposed the claim and supported it. That scholar gets one point added to the breakthrough component of their reputation.

For each claim in *BelievedFalse*, we find the first scholar who successfully refuted the claim. That scholar gets one point added to the breakthrough component of their reputation.

For each claim in *BelievedTrue*, but not in *BelievedOptimal*, we

find the first scholar who successfully strengthened the claim and supported the strengthened claim. That scholar gets amount of strengthening in points added to the breakthrough component of their reputation.

3.3 Histories and Open Publication

By publishing number of times a particular claim gets supported and refuted and the reputations of players supporting or refuting these claims, SCG creates social welfare.

When a lab gets into a stable state (a sub-optimal equilibrium where breakthroughs cease to happen for particular period of time), it is time to publish the current, maybe imperfect refutation and support strategies. If the scholars are avatars, their software is published. If the scholars are humans, an informal description of their techniques is published.

This levels the playing field and sets the stage for the next advancements. It is important to reward the scholars for their previous investment and not force them too early to publish their techniques.

4. DISCUSSION

In this section we show how binary SCG can be seen from the prespective of current game design models.

4.1 Five Element Game Ontology

Using the five element game ontology by Jose Zagal and Michael Mateas et al. [**?**]: interface, rules, goals, entities, and entity manipulation to give an overview of the SCG. The **entities** manipulated by the scholars are instances in *Instance*, solutions in *Solution* and claims in *Claim* in the context of a *Lab* and the reputation (payoff points) that a scholar accumulates. The **interface** for a scholar consists of the actions: *propose* a claim, decide to *refute*, *strengthen* or *agree* with a claim, to *provide* an instance and to *solve* an instance. The **rules** of the SCG are the rules of the Scientific Method: you must follow the refutation approach defined by *Lab*. You also must follow the scientific discourse prescribed by the extensive form representation of the game. The **goal** of the game is to make the refutation protocol predicate true or false, depending on the role you play (proponent or opponent of a claim). The **entity manipulation** includes solving instances with a certain quality and creating claims and instances. Gameplay is segmented into binary games and competitions and into increasingly more complex claims which become harder and harder to defend.

4.2 Triadic Game Design

Using the model of Triadic Game Design [**?**] by Casper Harteveld, we break down the SCG into a Reality, Meaning and Play component. Triadic Game Design describes an approach to serious game design and the SCG defines a large family of serious games. The **Reality** component of the SCG models a scientific community based on the Popper-style Scientific Method [**?**]. The **Meaning** component consists of cleaning the knowledge base of false or non-optimal claims by developing new constructions which are better than the constructions of others. The **Play** component of the SCG consists of competition, collaboration and finding a treasure (a new construction).

4.3 Exploratory-Performatory Games

According to Jonas Linderoth [**?**] games challenge two aspects of human nature: our ability to choose appropriate actions and our ability to perform appropriate actions. [**?**] views gaming as a cycle between interrelated exploratory and performatory actions.

What are the exploratory and performatory actions in the SCG? **Exploratory** actions are: (1) proposing a claim which means choosing from a set of claims. (2) Choosing an action: refute, agree or strengthen a given claim. **Performatory** actions are: (1) the proponent should defend the proposed claim. (2) If an opponent decides to refute, he should succeed in refuting, etc. In the SCG we also have a cycle of interrelated exploratory and performatory actions. Indeed, this cycle is the dominant activity in the SCG.

5. BINARY SCG ANALYSIS

In this section, we characterize blameable moves in the binary SCG game tree. We argue that blame is consistent with the design goal of encouraging scholars to put as much effort and intelligence as possible into proposing claims that are hard to falsify as well as into disputing those claims. We show that the payoff is sound, fair, and competitive with respect to the blameable actions. These three properties are important for SCG to be interesting. For space reasons, we deal in this subsection only with bivalent labs.

5.1 Blame Assignment

First, we divide the branches of the SCG game tree into 8 different sets based on 3 properties. Then, we assign blame to these sets. Table 3 presents summarizes these sets. The three properties are: (1) whether the proponent proposed a true claim or not. This is represented in Table 3 by the column **claim**. A value of `T` means the proposed claim is true. A value of `F` means the proposed claim was false. (2) the decision made by the opponent. This is represented in Table 3 by the column **dec**. A value of `a` means agree and a value of `d` means dispute. There is no strengthening in bivalent labs. (3) the outcome of the refutation game and the winner. This is represented in Table 3 by the column **out**. The outcome is either `s` for “support" or `r` for “refutation” and the winner is either `P` for the “proponent” or `O` for the “opponent”.

Table 3 has three blame columns **cB**, **aB** and **oB** with values that are either `P` for the “proponent” or `O` for the “opponent”. In the **cB** column, we blame the proponent for proposing a false claim. In the **oB**, we blame the opponent for either agreeing with a false claim or disputing a true claim.

The definition of those properties depends on the SCG CSP Table (see Figure 3 with the competitive payoff function). which analyses the SCG game tree (see Figure 1 on page 3 with the competitive payoff function). The SCG CSP Table has 8 rows based on the 3 independent variables: **claim** (true/false) abbreviated (T/F), **dec** (dispute/agree) decision abbreviated (d/a) and **out** (refuted/supported) outcome abbreviated (r/s). The claim variable value we would like to determine by playing the game. The dec (decision) variable is assigned by the second game decision. The out (outcome) variable is assigned by the refutation game where the claim is either refuted or supported. For the outcome we also list the winner of the game: sO means the claim was supported and the winner is O.

The SCG CSP Table has besides the three independent variables claim, action and outcome, three dependent blame variables **cP**, **aB** and **oB** with values P (proponent) or O (Opponent). Blame is is assigned as shown in the Blame Justification column of the SCG CSP Table. The blame justification (last column) is for the oB column only and the justification is independent of whether the claim is true or false. The justification is based on the observable variables dec and out. The justification for the cB column is: Proposing a false claim leads to a blame for P in column cB. For the decision blame column aB, agreeing with a false claim or disputing a true claim leads to a blame for O.

5.2 Payoff

5.2.1 Incomplete Knowledge

If one of the player is to blame in all rows of row group, then the payoff must be -ve. If one of the player is not to blame in any of the rows of a row group, then the payoff must be +ve.

In the design of the SCG payoff function there are three forces at work that need to be balanced. We would like the game to be fair, sound and competitive. The definition of those properties depends on the SCG CSP Table (see Figure 3 with the competitive payoff function). which analyses the SCG game tree (see Figure 1 on page 3 with the competitive payoff function). The SCG CSP Table has 8 rows based on the 3 independent variables: **claim** (true/false) abbreviated (T/F), **dec** (dispute/agree) decision abbreviated (d/a) and **out** (refuted/supported) outcome abbreviated (r/s). The claim variable value we would like to determine by playing the game. The dec (decision) variable is assigned by the second game decision. The out (outcome) variable is assigned by the refutation game where the claim is either refuted or supported. For the outcome we also list the winner of the game: sO means the claim was supported and the winner is O.

The SCG CSP Table has besides the three independent variables claim, action and outcome, three dependent blame variables **cP**, **aB** and **oB** with values P (proponent) or O (Opponent). Blame is is assigned as shown in the Blame Justification column of the SCG CSP Table. The blame justification (last column) is for the oB column only and the justification is independent of whether the claim is true or false. The justification is based on the observable variables dec and out. The justification for the cB column is: Proposing a false claim leads to a blame for P in column cB. For the decision blame column aB, agreeing with a false claim or disputing a true claim leads to a blame for O.

Finally, the **P** and **O** columns give the payoff values for the proponent P and opponent O. Notice that the payoff function has identical values for row pairs (1,2) and (3,4) and (5,6) and (7,8).

A game is **fair** if for all rows the following property holds: If P (or O) lose a point they are blamed in that row. See Figure 3 for an example of a fair game.

Note that if P could have made a mistake, but we are not sure because we don't know whether the claim is true or false, we don't punish P. Therefore, if P made a point, it could still be blamed. We give P the benefit of the doubt.

The soundness definition has two parts: Definitive soundness and chance soundness. The first kind of soundness is about the oB column: if you get blamed for losing a refutation game you will have a negative payoff. The second kind of soundness is about the cB and aB columns: if you get blamed in those columns you take the risk of having a negative payoff. It depends on your opponent whether she will expose the reason for the blame.

A game is **oB-sound** if for all rows the following property holds: if P (or O) are blamed in the oB column, they have a negative payoff in that row. The game in Figure 3 is oB-sound.

A game is **cB-sound** if there exists a row in which P is blamed in the cB column and there is a negative payoff for P. The game in Figure 3 is cB-sound. Proof: If P is blamed in column cB, the optimal response for O is to dispute the claim and then to refute it. Row 7 has a negative payoff for P.

Informally, cb-sound means that if you get blamed in the cB column you risk of being punished by a strong opponent.

A game is **aB-sound** if for all rows where O is blamed in the aB column, there is a chance that O will have a negative payoff if P plays better. The game in Figure 3 is aB-sound. The proof involves two cases. The first case is: O makes the mistake of agreeing with a false claim (F a, row 1 and row 5). We search for a row with the same prefix (F a) where P is the winner. We find row 5 where the payoff is indeed negative for O. The second case is: O makes the mistake of disputing a true claim (T d, row 4 and row 8). We search for a row with the same prefix (T d) where P is the winner. We find row 4 where the payoff is indeed negative for O.

A game is **sound** if it has all three properties: oB-sound and cB-sound and aB-sound.

The third important game property is competitiveness. A game is **competitive** if it matters whether you win or lose: the payoff value is higher for the winner. The game in Figure 3 is competitive.

Theorem: Good Payoff There exists a payoff function which has all three properties: fair, sound and competitive.

Proof: The game in Figure 3 is fair, competititive and sound.

claim	dec	out	P	O	cB	aB	oB	Blame Justification
F *	a	sO	*paso*	*oaso*	P	O	-	
T	a	sO			-	-	-	
F *	d	sP	*pdsp*	*odsp*	P	-	O	O did not refute a claim it disputed
T	d	sP			-	O	O	
F	a	rP	*parp*	*oarp*	P	O	O	O failed to support a claim it agreed with
T *	a	rP			-	-	O	
F	d	rO	*pdro*	*odro*	P	-	P	P failed to support a claim it proposed
T *	d	rO			-	O	P	

Figure 3: SCG CSP Table.

5.3 Payoff Function Design using CSP

To study the space of payoff functions, we introduce a constraint satisfaction problem that expresses the constraints implied by the fair, sound and competitive properties. Figure 3 introduces eight variables for formulating the constraints for the payoff function: $paso$, $oaso$, etc. The variable names come from the decision and outcome column.

The constraints for the fair property (losing a point implies a blame in oB) are: $paso \geq 0$, and $oaso \geq 0$, and $pdsp \geq 0$, and $parp \geq 0$, and $odro \geq 0$.

The constraints for soundness we give separately for oB-sound, cB-sound and aB-sound.

For oB-sound the constraints are: $odsp < 0$, and $oarp < 0$, and $pdro < 0$.

For cB-sound the constraint is $((pdro < 0) \vee (parp < 0) \vee (pdsp < 0) \vee (paso < 0))$.

For aB-sound the first constraint is $((oaso < 0) \vee (oarp < 0))$ because row 1 or 5 must have a negative payoff for O. The second constraint is $((odsp < 0) \vee (odro < 0))$ because row 4 or 8 must have a negative payoff for O.

The constraints for competitiveness are (good to win the refutation game): $parp - oarp > paso - oaso$ and $pdsp - odsp > pdro - odro$. The rationale is as follows: In case of agreement, the difference between the payoff of the proponent and the payoff of the opponent when the proponent wins $(parp - oarp)$, is larger than the corresponding difference when the opponent wins $(paso - oaso)$. Therefore, wining the refutation game is advisable

for both parties. The second constraint applies for the case of dispute.

Any variations of the payoff function must satisfy all constraints introduced above if the game has to be fair, sound and competitive. The above constraints are also useful when only a subset of the properties is desired.

6. REFERENCES