The Scientific Community Game

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

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

10/15/2012

ABSTRACT
We provide a clean definition of the Scientific Community Game (SCG, formerly called the Specker Challenge Game) and prove basic properties. An example illustrates the concepts.

The key contribution of this paper is a simple protocol language which is at the heart of defining constructive claims through games. Our protocol language generalizes the class of claims that can be expressed in predicate logic. We build on the long tradition in logic and games of imperfect information: our protocol language can also express indeterminate claims, such as "I am better than you at solving this problem."

The purpose of SCG is to provide a helpful framework to develop and disseminate the world's constructive claims in formal scientific domains. The development of claims is both collaborative and self-evaluating using the global brain.

INTRODUCTION
The Scientific Method has not yet been implemented as a generic software tool that can be customized for various domains to develop new knowledge. We present our design of such a generic tool. Applications range from research (refuting and defending claims) to teaching (traditional, online and massively open online) courses in STEM areas. While this paper defines SCG, there are separate papers planned that describe various applications of SCG.

- SCG for Funding Agencies

  Funding agencies, such as NSF, define, in collaboration with interested researchers, labs that define the problem to be solved. The self-evaluating nature of SCG will fairly evaluate the contributions of the scholars and the collaborative nature will lead to productive team work. Newcomers can contribute by participating in a long-running lab (dozens of years).

- SCG for Software Development

  A computational task is defined by a lab where the role of scholar is played by an avatar (software). Competitions are held and the winning avatars will contain the best algorithms for the computational task.

- SCG for Teaching

  To teach a specific skill, we design a lab which encourages scholars to acquire that skill to perform well in the lab. For teaching it is important to use a payoff function which reduces the competitiveness of the game and which enhances the collaborative nature. Good students become effective teachers to the weaker students. Our experience has been that optimization labs work best for teaching.

We focus on formal sciences although the idea applies to both formal sciences and other sciences. Unlike other sciences, the formal sciences are not concerned with the validity of claims based on observations in the real world, but instead with the properties of formal systems based on definitions and rules. Examples of formal sciences are: logic, mathematics, theoretical computer science, information theory, systems theory, decision theory, statistics. Natural sciences for which adequate simulators exist, become formal sciences.

In SCG, claims are organized into labs where they are exposed to refutation. A refutation attempt initiates a scientific dialog between proponent and opponent which induces collaborative behavior. The labs are self-evaluating in that there is no need of a third party to evaluate the contributions of the scholars: they evaluate each other fairly. This is enforced by the SCG-level generic game rules and refined by the lab-level more specialized rules.

We have developed an implementation of SCG which is available on SourceForge [1]. The following SCG definition is an abstraction of our implementation.

SCG users
There are two levels of SCG users: Lab designers and scholars. SCG defines generic rules for a scientific community and lab designers specialize those rules to a specific domain which in turn defines the tasks to be performed by the scholars.

- Lab designers have a computational problem they want to have solved. The role of lab designer can be played

Submitted for review to Foundations of Digital Games, 2013, Crete.

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 and as return on their investment they get a lab where scholars are invited to participate. Defining a lab is like writing a program for the global brain [5].

- 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.

BINARY GAME DEFINITION

We first define the SCG binary game which is the building block for competitions. We define the SCG binary game through its extensive-form (or tree-form) representation which is a common representation in game theory [11].

We call the players scholars because the game is about making scientific or technical claims and defending them.

The extensive-form representation of SCG binary game is given by: (1) Two scholars, called Proponent (female) and Opponent (male). (2) For both scholars, every opportunity they have to move. The sequencing of moves is important. (3) What each scholar can do for each of their moves. They can choose from some set, e.g., claims, instances and solutions or make a decision about classifying claims. (4) what each scholar knows for every move. (5) the payoffs received by both scholars for every possible combination of moves.

To define the moves, we need a few sets and functions which make up what we call a lab where experiments and measurements may be performed in a specific domain. A lab is an environment where (human) computation solves a problem and develops a knowledge base of claims. By a problem we mean the set of instances in *Instance*. The lab definition consists of *Instance*, *Solution*, *Claim*, *valid*, *quality*, *stronger* which are defined next.

Instance and Solution

Instance is defining a set of instances to be solved. The definition of an *Instance* is split into two parts: the parameters which constrain a family of instances and the structure which defines the structure of the instances based on specific values of the parameters.

Solution is defining a set of solutions. We need a *valid* function to check whether a solution $s \in Solution$ is valid for an instance $i \in Instance$:

$$\text{boolean } valid(i, s).$$

We need a *quality* function to return a real number for a given instance $i \in Instance$ and solution $s \in Solution$:

$$\text{float } quality(i, s).$$

Claim

The first choice made in the SCG binary game tree of the extensive form representation is to choose a claim from *Claim*. A claim consists of a definition of *Instance* and *Solution*, a protocol and a rational number. The rational number has the meaning of a quality that can be achieved when solving a constructive task that precisely defines the allowed inputs ($i \in Instance$) and the allowed solutions ($s \in Solution$ satisfying *valid(i,s)*) with $quality(i, s)$. In an earlier version of Scientific Community Game we used the term "challenge" instead of "claim." A claim challenges you to refute it.

We need a function to compare claims. Function

$$stronger(c_1, c_2)$$

checks for two claims c_1, c_2 in *Claim* whether claim c_1 is stronger than claim c_2. For some labs, *stronger* always returns false.

The second choice in the SCG binary game tree is of a different nature: to decide whether (to try) to *refute*, *strengthen*, or *agree* with the claim. Depending on the choice made, the refutation protocol is executed with the appropriate arguments. *strengthen* and *agree* are also reduced to *refute*.

Refutation protocol

The refutation protocol decides one's performance when interacting with a scholar in a given lab. It provides opportunities for learning and trying to figure out how the other scholar "thinks". It defines the scientific discourse to be used to "test" a claim.

The refutation protocol has three arguments: a claim c and a Proponent and an Opponent and consists of a list of steps followed by a refutation predicate which receives the data as arguments. The list of steps defines data (instances in *Instance* and solutions in *Solution*) that are given to the refutation predicate.

Protocols are defined by the following grammar: (// is the comment symbol)

```
ProtocolSpec =  List(Step).
Step =
  Action "from" Role.
interface Role = Proponent | Opponent.
Proponent = "Proponent".
Opponent = "Opponent".
interface Action =
  ProvideAction | SolveAction.
ProvideAction = "instance".

// solve the instance provided in
// step # stepNo.
// stepNo is 0-based.
SolveAction = "solution" "of" // stepNo
```

```
int.
```

Now that the claim definition is complete, we define a claim to be true if it has a defense 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 will always succeed in refuting. We define a claim to be indeterminate if it has neither a defense nor a refutation strategy.

Consider the following protocol instance used in a claim proposed by Proponent:

```
instance from Proponent // i
solution of 0 from Opponent // sB
```

The protocol predicate is $p(i, sB, claim)$.

Proponent expresses the claim: There exists an instance *i* in *Instance* so that for all solutions *sB* of *i* for which the predicate *valid(i,sB) is true, the predicate p(i,sB,claim)* holds. In order to support her claim, the Proponent has to deliver i and Opponent has to deliver *sB* in a resource constrained environment.

If we add the following line to the end of the above protocol

```
solution of 0 from Proponent // sA
```

and define the protocol predicate to be:

$$p(i, sA, sB, claim) = (quality(sA, i) \geq quality(sB, i)),$$

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.

As a general rule, solutions are all kept secret until protocol evaluation. When Proponent computes *sA* she does not know *sB*.

In SCG, the payoff is determined by the value of a refutation predicate which has as parameters the values collected along the current path back to the root. The value of the refutation predicate determines the payoffs.

The game tree is given by:

- Proponent: propose claim *c* from *Claim*.
- Opponent: decide between 3 alternatives:
  - *refute*,
  - *strengthen* with a stronger claim *c'* from *Claim*, i.e., $stronger(c', c)$,
  - agree.

Function *refute* directly invokes the refutation protocol

$$refute(c, \text{Proponent}, \text{Opponent}).$$

Function *strengthen(c')* invokes the refutation protocol

$$refute(c', \text{Opponent}, \text{Proponent})$$

with the scholars switched. Function *agree* invokes the refutation protocol

$$refute(c, \text{Opponent}, \text{Proponent})$$

also with the scholars switched.

The payoff information is determined by the refutation predicate $p(c, ...)$. The three dots ... refer to the data collected by the refutation protocol, consisting of a heterogeneous sequence of instances and solutions. The Proponent tries to make the predicate true while the Opposer tries to make it false. The refutation is successful when the predicate is false.

We use the following notation:

$$p(c, ...)?(\text{Pt}, \text{Ot}), (\text{Pf}, \text{Of}).$$

If $p(c, ...)$ is true, Pt is the payoff for the Proponent and Ot is the payoff for the Opponent. If $p(c, ...)$ is false, Pf is the payoff for the Proponent and Of is the payoff for the Opponent.

We give an example of a payoff function, called competitive payoff (zero-sum), which is the default pay-off function:

1. refute: $p(c, ...)?(1, -1) : (-1, 1)$

   If the predicate is true, the Proponent gets rewarded with one point because she successfully defended her claim. If the predicate is false, the Opponent has won and gets a point.

2. strengthen c to c': $p(c', ...)?(-1, 1) : (1, -1)$

   If the predicate is true, the Opponent gets rewarded with one point because he successfully defended the stronger claim. If the predicate is false, the Proponent has won and gets a point. The strengthening was not successful.

3. agree: $p(c, ...)?(0, 0) : (1, -1)$

   If the predicate is true, the Opponent has successfully defended the claim and nobody gets a point. Agreement is successful. If the predicate is false, the Opponent has failed to defend the claim and loses a point. Agreement is not successful.

Depending on the application, many more meaningful payoff functions can be defined. For example, if SCG is used for creating student interaction in a MOOC, I recommend the following low competition payoff function:

1. refute: $p(c, ...)?(0, 0) : (0, 1)$

   If the predicate is true, nobody gets a point because we want the Opponent to learn from the Proponent through the refutation protocol. If the predicate is false, the Opponent has won and gets a point.

2. strengthen c to c': $p(c', ...)?(0, 1) : (0, 0)$

   If the predicate is true, the Opponent gets rewarded with one point because he successfully defended the stronger claim. If the predicate is false, the Proponent has won but does not get a point because we want the Opponent to have cheap opportunities to attack and learn.

3. agree: $p(c, ...)?(0, 0) : (1, 0)$

   If the predicate is true, the Opponent has successfully defended the claim and nobody gets a point. If the predicate is false, the Opponent has failed to defend the claim but has gained information to learn. The Proponent earns a point.

The competitive payoff and the low competition payoff are two examples of payoff functions that promote good behavior in the lab. Other payoff functions are possible.

Instances are only available when they are needed. For example, in the spirit of the Renaissance mathematical competitions between Tartaglia and Fior, if the protocol asks that the Proponent and Opponent deliver each 10 instances, followed by the solution activity. The instances are secret until they are solved.

Rules

The rules for the SCG binary game are natural: follow the extensive-form representation of the game definition. A scholar loses immediately, if he or she delivers an element that is not in the specified set, or if he or she does otherwise not follow the refutation protocol.

All moves must be completed within the given resource (time, space, etc.) limit.

The winner of a binary game is the scholar which achieves the highest payoff. The scholars are egoistic but they will create social welfare (a knowledge base) when we engage them in a competition.

LABS

Labs play a central role in SCG. They define a domain to be investigated and require great care in their definition. A lab definition can be viewed as a declarative program to define a human computation [5].

We distinguish between standard labs and optimization labs.

- Standard lab

  Claims in a standard lab are either true, false or indeterminate. The *stronger* relation for claims is always false.

- Optimization lab

  In an optimization lab we have a family of claims parameterized by quality: $C(q), q \in [0, 1]$. The *stronger* relation is based on comparing qualities. Bivalence holds: a claim is either true or false.

  An optimization lab may contain a union of such families of optimization claims.

Failure of bivalence: imperfect information

SCG can express indeterminate claims that are neither true nor false. Such claims were studied in Independence Friendly Logic [20], an extension of first-order logic.

Consider the following lab: *Instance* = positive real numbers. *Solution* = real numbers. The protocol

```
instance from Proponent
solution of 0 from Opponent //sO
solution of 0 from Proponent //sP
```

The predicate is: $sO = sP$. According to the SCG rules, sO is not known when sP is computed. The lab contains only one claim where the quality is irrelevant. This claim is neither true nor false: bivalence fails. Notice the similarity to the "at least as good as" claim discussed earlier.

Therefore we also distinguish between bivalent labs (all claims are either true or false) and non-bivalent labs (some claims are indeterminate).

Open publication

When a lab gets into a stable state (a sub-optimal equilibrium), it is time

Scholar role

So far we used the generic scholar role to talk about lab design. Some labs are simple enough to expect that there is an algorithm to act as a scholar: to propose and oppose claims and to provide instances and solve instances during the refutation protocol.

Therefore, two instantiations of SCG: SCG Human and SCG Avatar. In SCG Human, a human will control the propose and oppose activity, maybe by using a program to facilitate the task. In SCG Avatar, a program will control the propose/oppose activity. In SCG Avatar the primary interest is in the solve function. The winning avatar is likely to have the better solve function, although providing difficult instances is also important to winning.

- SCG Human

  For hard problems. An algorithm is not known (yet).

  We have to be careful with the time of the scholars.

  The knowledge is in the head of the scholars.

- SCG Avatar

  For problems that are solvable by an algorithm.

  The scholars are cheap and can work all night when the cloud is not busy.

  The knowledge is encoded in the avatar software.

We find SCG Avatar to be a useful software development process model to develop software for computational tasks using competitions.

COMPETITIONS

For SCG Avatar it makes sense to organize full round-robin competitions where each avatar plays against each other avatar using a binary game. If the lab is inhabited by n avatars, we get $n \cdot (n-1)$ binary games. The avatars are ranked according to their accumulated payoff. The highest ranked avatar most likely contains the best solve method which can be further polished and put into a product.

For SCG Scholar it makes sense to organize Swiss style competitions which require fewer binary games than full round-robin competitions.

The competitions will create a knowledge base of claims that are mostly defended. This is the social welfare created by the egoistic scholars.

Open publication

When a lab gets into a stable state (a sub-optimal equilibrium), it is time to publish the current, maybe imperfect refutation and defense 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. But it is important to reward the scholars for their previous investment and not force them too early to publish their techniques.

Timing

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 being-the-first 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: defend, refute, strengthen. 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 defended it. TimingReputation +1.

For each claim in BelievedFalse, we find the first scholar who refuted the claim. TimingReputation +1.

For each claim in BelievedTrue, but not in BelievedOptimal, we find the first scholar who strengthened the claim and defended the strengthened claim. TimingReputation +1 + amount of strengthening.

The reputation of a scholar consists of two components: The payoff points collected and the timing reputation points collected.

EXAMPLE

We give an example of lab definitions to illustrate the concept. We developed a Lab Designer Guide [13] which contains many more examples of labs (formerly called playgrounds). A widely applicable way to define a lab is to use alternating quantifier predicate logic expressions, such as ForAllExists or ExistsForAll. Consider the fundamental theorem of arithmetic, which states that every integer greater than 1 is either prime itself or is the product of prime numbers. This claim is of the form ForAllExists (for all non-prime natural numbers there exists ...) and it is easy to create a lab to practice the fundamental theorem of arithmetic.

Worst-case input lab

The goal of this lab is to find worst-case inputs to an algorithm. As a concrete example, we use the Gale-Shapley algorithm (GS) for stable matchings. GS consists of one while loop and the goal is to find the worst-case input for n men and n women that produces the most number iterations.

Instance: The parameter is n = number of men = number of women. An instance contains the number of executions of the while-loop for a worst-case input.

Solution: An input to Gale-Shapley (the set of preferences for men and women).

valid(i,s): the GS input s achieves the desired number of iterations i.

quality(i,s): the number of iterations of the while loop generated by s.

Claim: a claim has the form GS(n,u), where n is the number of men (and women) and u is the number of iterations for worst input for n men and women. stronger(c1,c2): if c1 has more iterations than c2.

GS(10,30) says that for 10 men and women, there is an input that achieves 30 iterations. GS(10,40) is a stronger claim.

The protocol is:

```
0 instance from Proponent
  //iP: iterations
1 instance from Opponent
  //iO: iterations
2 solution of 0 from Proponent
  //GS input creating iP iterations
3 solution of 1 from Opponent
  //GS input with iO iterations
```

The protocol predicate is $iP > iO$. The rationale for this protocol design is that the GS inputs need to be hidden from the two competitors until the game is over. The reason is that seeing the worst GS input would reveal too much information to the other party. In SCG, solutions are only revealed at the end of a binary game. Note that each scholar solves its own instance.

PROPERTIES

We list basic properties of SCG. The first two are formal and we give a short proof.

Egalitarian

When you make a mistake, you might be caught.

There are 7 different mistakes (one per line, the * lines are for optimization labs):

```
P = Proponent
```

```
O = Opponent

propose(P,c),c=false
 * propose(P,c),c=not optimum, c=true
refute(P,O,c),c=true
 * strengthen(P,O,c,cs),c=optimum
strengthen(P,O,c,cs),c=false
agree(P,O,c),c=false
 * agree(P,O,c),c=not optimum, c=true
```

Theorem [Egalitarian] For all labs and for mistake made in a lab there exists a reaction that leads to a negative payoff for the competitive payoff function.

For formal sciences, SCG is not sound in the following sense: It is possible that during a binary game faulty decisions are made, e.g., a true claim is refuted or a false claim is defended. But SCG has the property that faulty decisions might lead to a negative payoff (under the competitive payoff function). If a faulty action was made there exists an exposing reaction that blames the bad action.

The proof of the egalitarian theorem is by case analysis. It is based on the following facts: For a true claim there exists a strategy to defend it. For a false claim there exists a strategy to refute it.

Of course, there are many other "faulty" moves but they are all caught by the rules of SCG and the violating scholar has lost.

For each of the seven cases there is a path in the game tree that leads to a negative payoff for the cholar who made the faulty decision. In SCG it is impossible that a strong scholar is ignored by other scholars as it might happen in a real scientific community.

Convergence

There is a second simple property of SCG: When no mistakes are made in an optimization lab, the optimum claim will eventually be found.

Theorem [Convergence]: Consider a set C of claims $c(t)$, where t is rational between 0 and 1. $c(0)$ is true and $c(1)$ is false and there is an optimal value t_0 of t where the truth value of $c(t_0)$ switches from true to false. If a sequence of binary games is played using claims in C and binary search without mistakes, the optimal claim $c(t_0)$ will be found.

Equilibria

A lab can have several equilibria which correspond to increasingly better levels of knowledge.

Social production

Social Production is the creation of artefacts, by combining the input from contributors with weak ties without predetermining the way to do this. Integration of new and innovative contributions is achieved by trading winning for bug reports that contradict the claims being made. The innovative contributions are kept secret to foster competition. When you win you must release a bug report that might give a hint about the clever approach you use.

EXPERIENCE WITH THE Scientific Community Game

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

Software Development

The most successful graduate classes were the ones that developed and maintained the software for SCG Court [1] as well as several labs and their avatars to test SCG Court. Developing labs for avatars has the flavor of defining a virtual world for artificial creatures. At the same time the students got detailed knowledge of some problem domain and how to solve it. A fun lab was the Highest Safe Rung lab from [10] where the best avatars need to solve a constrained search problem using a modified Pascal triangle.

Algorithms

The most successful course (using [10] as textbook) was in Spring 2012 where the interaction through SCG encouraged the students to solve difficult problems. Almost all homework problems were defined through labs and the students posted both their exploratory and performatory actions on piazza.com. We used a multiplayer version of the SCG binary game which created a bit of an information overload. Sticking to binary games would have been better but requires splitting the students into pairs. The informal use of SCG through Piazza (piazza.com) proved successful. All actions were expressed in JSON which allowed the students to use a wide variety of programming languages to implement their algorithms.

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

Without SCG I don't believe that the students would have created the same impressive results. SCG effectively focuses the scientific discourse on the problem to be solved.

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

SCIENTIFIC COMMUNITY

The informal model behind our scientific communities is described by the following list. Scholars are encouraged to (1) propose claims that are not easily improved. (2) propose claims that they can successfully defend. (3) quote related claims and show how they improve on previous claims. (4) prove claims if the current state of the art allows. (5) stay active and propose new claims or oppose current claims. (6) be well-rounded: solve posed instances and pose difficult instances for others. (7) become famous! (maximizing their payoff and be the first to propose and defend a true or optimal claim)

RELATED WORK

SCG has not grown in a vacuum. We make connections to several related areas.

Crowd Sourcing and Human Computation

There are several websites that organize competitions. What is common to many of those competitions? We believe that SCG provides a foundation to websites such as TopCoder [21] or the more specialized kaggle.com.

SCG makes a specific, but incomplete proposal of a programming interface to work with the global brain [5]. What is currently missing is a payment mechanism for the scholars and an algorithm to split workers into pairs based on their background.

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

Scientific discovery games, such as FoldIt and EteRNA, are variants of SCG. [6] describes the challenges behind developing scientific discovery games. [3] argues that complex games such as FoldIt benefit from tutorials. This also applies to SCG but a big part of the tutorial is reusable across scientific disciplines.

Logic and Imperfect Information Games

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

Logical Games [18], [7] have a long history. SCG is an imperfect information game builds on Paul Lorenzen's dialogical games [9].

Foundations of Digital Games

According to Jonas Linderoth [17] games challenge two aspects of human nature: our ability to choose appropriate actions and our ability to perform appropriate actions. [17] views gaming as a cycle between interrelated exploratory and performatory actions.

What are the exploratory and performatory actions in 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) a claim for which the refute action was chosen should refute the claim, etc. In SCG we also have a cycle of interrelated exploratory and performatory actions.

A functioning game should be deep, fair and interesting which requires careful and time-consuming balancing. [8] describes techniques used for balancing that complement the expensive playtesting. This research is relevant to SCG lab design. For example, if there is an easy way to refute claims without doing hard work, the lab is unbalanced. SCG has the interesting property that if a "clever procedure" is know to scholar Alice, but not to scholar Bob, Alice can easily win over Bob. If both know the clever procedure, the game is a tie. The goal of an SCG lab is to discover such a clever procedure.

Architecting Socio-Technical Ecosystems

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

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

Online Judges

An online judge is an online system to test programs in programming contests. A recent entry is [19] where private inputs are used to test the programs. [21] includes an online judge capability but where the inputs are provided by competitors. This dynamic benchmark capability is also expressible with SCG: The claims say that for a given program that all inputs create the correct output. A refutation is an input which creates the wrong result.

Educational Games

SCG can be used as an educational game. One way to create adaptivity for learning is to create an avatar that poses gradually harder claims and instances. Another way is to pair the learner with another learner who is stronger. [2] uses concept maps to guide the learning. Concept maps are important during lab design: they describe the concepts that need to be mastered by the students for succeeding in the game.

Origins of the Scientific Community Game

A preliminary definition of SCG was given in a keynote paper [14]. [12] gives further information on the Scientific Community Game.

The original motivation for SCG came from the two papers with Ernst Specker: [15] and the follow-on paper [16].

Renaissance competitions: the public problem solving duel between Fior and Tartaglia, about 1535, can easily be expressed with the SCG protocol language.

SUMMARY AND CONCLUSIONS

SCG provides a simple interface to a scientific community that uses the Scientific Method. SCG provides for effective customization of the generic scientific machinery by using lab definitions. Since SCG models a scientific community it is widely applicable and deserves a central place in the world's cyberinfrastructure.

REFERENCES

1. A. Abdelmeged and K. J. Lieberherr. SCG Court: Generator of teaching/innovation labs. Website, 2011. http://sourceforge.net/p/generic-scg/code-0/110/tree/GenericSCG/ .
2. E. Andersen. Optimizing adaptivity in educational games. In *Proceedings of the International Conference on the Foundations of Digital Games*, FDG '12, pages 279–281, New York, NY, USA, 2012. ACM.
3. E. Andersen, E. O'Rourke, Y.-E. Liu, R. Snider, J. Lowdermilk, D. Truong, S. Cooper, and Z. Popovic. The impact of tutorials on games of varying complexity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 59–68, New York, NY, USA, 2012. ACM.
4. J. Attenberg, P. Ipeirotis, and F. Provost. Beat the machine: Challenging workers to find the unknown unknowns. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.
5. A. Bernstein, M. Klein, and T. W. Malone. Programming the global brain. *Commun. ACM*, 55(5):41–43, May 2012.
6. S. Cooper, A. Treuille, J. Barbero, A. Leaver-Fay, K. Tuite, F. Khatib, A. C. Snyder, M. Beenen, D. Salesin, D. Baker, and Z. Popović. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 40–47, New York, NY, USA, 2010. ACM.
7. W. Hodges. Logic and games. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2009 edition, 2009.
8. A. Jaffe, A. Miller, E. Andersen, Y.-E. Liu, A. Karlin, and Z. Popovic. Evaluating competitive game balance with restricted play, 2012.
9. L. Keiff. Dialogical logic. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2011 edition, 2011.
10. J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.
11. K. Leyton-Brown and Y. Shoham. Essentials of game theory: A concise multidisciplinary introduction. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2(1):1–88, 2008.
12. K. Lieberherr. The Scientific Community Game. Website, 2009. http://www.ccs.neu.edu/home/lieber/evergreen/specker/scg-home.html.
13. K. J. Lieberherr. Scientific Community Game Lab Designer Guide. Website, 2011. http://www.ccs.neu.edu/home/lieber/courses/se-courses/cs5500/sp11/projects/playground-designer-user-guide.html.
14. K. J. Lieberherr, A. Abdelmeged, and B. Chadwick. The Specker Challenge Game for Education and Innovation in Constructive Domains. In *Keynote paper at Bionetics 2010, Cambridge, MA, and CCIS Technical Report NU-CCIS-2010-19*, December 2010. http://www.ccs.neu.edu/home/lieber/evergreen/specker/paper/bionetics-2010.pdf .
15. K. J. Lieberherr and E. Specker. Complexity of Partial Satisfaction. *Journal of the ACM*, 28(2):411–421, 1981. http://www.ccs.neu.edu/home/lieber/p-optimal/JACM1981.pdf .
16. K. J. Lieberherr and E. Specker. Complexity of Partial Satisfaction II. *Elemente der Mathematik*, 67(3):134–150, 2012. http://www.ccs.neu.edu/home/lieber/p-optimal/partial-sat-II/Partial-SAT2.pdf.
17. J. Linderoth. Why gamers don't learn more: An ecological approach to games as learning environments. In L. Petri, T. A. Mette, V. Harko, and W. Annika, editors, *Proceedings of DiGRA Nordic 2010: Experiencing Games: Games, Play, and Players*, Stockholm, January 2010. University of Stockholm.
18. M. Marion. Why Play Logical Games. Website, 2009. http://www.philomath.uqam.ca/doc/LogicalGames.pdf.
19. J. Petit, O. Giménez, and S. Roura. Jutge.org: an educational programming judge. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, pages 445–450, New York, NY, USA, 2012. ACM.
20. T. Tero. Independence friendly logic. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2011 edition, 2011.
21. TopCoder. The TopCoder Community. Website, 2009. http://www.topcoder.com/.