The Specker Challenge Game for Education and Innovation in Constructive Domains

Karl Lieberherr, Ahmed Abdelmeged, and Bryan Chadwick

College of Computer & Information Science
Northeastern University, 360 Huntington Avenue
Boston, MA, 02115 USA.
{lieber,mohsen,chadwick}@ccs.neu.edu

Dedicated to Ernst Specker to celebrate his 90th birthday.

1 Introduction

We have the following vision: To teach a constructive domain (e.g., domains in computer science, mathematics, engineering, etc.), we design a domain-specific game where the winning students demonstrate superior domain knowledge. The game is simple: the students make constructive claims about the domain we want to teach and refute each others' claims. The students who are the most successful in defending their claims and in refuting the claims of others win the game. Some benefits of the game are: (1) the students give each other constructive feedback about their claims. Students who lose points gain knowledge to improve their game in the future. (2) Although the students are egoistic, maximizing their points, they create a common good: knowledge. The claims that have not been successfully opposed are candidates for truth. (3) The game is fun and adjusts to the skill levels of the students. It can be played productively by undergraduates as well as by world-class researchers to push the state-of-the-art in a domain. The game also serves as a useful, and novel software development process to develop software for computational problems.

2 Claims and Refutation Protocols

At the heart of the game are two major concepts, those of *claims* and a corresponding *refutation protocol*. Claims are made by one student, Alice. Alice is then engaged with another student, Bob, in an instance of a refutation protocol to determine if Alice would be able to successfully support her claim. Or if Bob would be able to successfully refute Alice's claim. Or if both Alice and Bob will tie.

2.1 Claims in a Nutshell

We distinguish between mathematical and non-mathematical claims.

– Mathematical. Those claims can be written as a mathematical statement that is either true or false. For example, $\exists(c, n_0)\ s.t.\ \forall n \geq n_0\ :\ n^{1/2} \leq c * (log(n))^{10}$. For those claims we can determine in principle in advance that we have a winning strategy for supporting or refuting them.
– Non-Mathematical. Those claims take the form of "I can construct a problem, knowing a very good solution for it that you cannot beat". or "If you construct a problem, knowing a very good solution for it, I can beat your solution". For those claims we cannot guarantee logically that we will win. We must play out the game and the refutation protocol will decide. An example of a non-mathematical claim is the claim for the Maximum Independent Set problem shown below or the claims in the renaissance game.

2.2 Refutation Protocol in a Nutshell

The refutation protocol involves a message exchange between both students, Alice and Bob, and a trusted third party called the administrator. The administrator serves as a mediator for messages; Alice and Bob communicate only to the administrator and the administrator forwards their messages to the other party. The administrator also serves as a referee that decides who succeeds in the protocol.

The exact sequence of messages exchanged messages depends on the claim. We assume that all three parties have agreed on a claim made by Alice. If the claim was of the form $\exists x \in X\ s.t. \forall y \in Y(x) : predicate(x, y)$. Then Alice has to provide an $x \in X$ such that Bob cannot find a $y \in Y(x)$ that makes $predicate(x, y)$ false. On the other hand, if the claim was of the form $\forall x \in X\ \exists y \in Y(x) : predicate(x, y)$, then Bob has to provide an $x \in X$ such that Alice cannot find a $y \in Y(x)$ that makes $predicate(x, y)$ true. In general, a pair of messages needs to be exchanged per every pair of alternating qualifiers in the claim.

To handle non-mathematical claims, x may be extended to have an optional secret part that is used to hold the secret solution. The administrator does not pass the secret along to the other party, instead it stores it locally and uses it later in deciding on a winner.

Note that Alice is not required to prove her claim, neither is Bob required to disprove Alice's claim (unless the refutation protocol explicitly requires them to exchange proofs). Claims are supported or refuted via examples and counter examples. Therefore, if Alice successfully supports her claim (Alice wins), it does not mean that her claim is true. It could be that Bob made a mistake. If Bob successfully refuted Alice's claim (Bob wins), it does not mean that the claim is false. It could be that Alice made a mistake. However, we can state the following properties of the game: If Alice is perfect and Bob wins, the claim is false. If Bob is perfect and Alice wins, the claim is true. By "perfect" we mean making the best decisions to win the game.

2.3 Claim Design

Claim design must follow a few principles. Mathematical claims that come from interesting theorems with an even number of alternating quantifiers are good candidates for claims. This is folklore knowledge.

The claims must have the property that it is "challenging" for Alice to produce her x and for Bob to produce his y (knowing x). It must be "challenging" for Alice to produce claims that cannot be refuted. It must be "challenging" for Bob to analyze a claim to reach the refute decision. Another issue in claim design is privacy. The information that is exchanged between the players should not reveal too many secrets. Otherwise, the players are less motivated to develop new ideas.

Claim design for educational purposes may have an indirect flavor. For some simple claims, a significant amount of material needs to be mastered that is not at all mentioned in the claim formulation.

2.4 Examples

O Notation How can we engage our students in a constructive discussion about the big O notation. The instructor prepares for them a set of claims to choose from and Alice chooses the following claim: $n^{1/2} = O((log(n))^{10})$. The parameters to the refutation protocol are: $X = \mathbb{R}^+ \otimes \mathbb{R}^+$, where $\mathbb{R}^+$ is the set of positive real numbers, and $Y = \mathbb{R}^+$. There are no secrets in this case. The refute function is $\mathit{refute}((C, n_0), n) = (n < n_0) \vee (n^{1/2} > C \cdot ((log(n))^{10}))$. The refutation protocol is: Alice finds $x = (C, n_0)$ and Bob finds $n \geq n_0$ so that refute returns true. After playing this game a few times, Bob starts to refute consistently and they start looking for a reason. They remember a theorem from calculus: $log(n) \in O(n^x)$ for every $x > 0$.

Highest Safe Rung Problem How can we engage our students in a constructive discussion about search, from linear search to binary search and what is between? Along the way, we want to teach them about binary decision trees and attempting to minimize the longest root-to-leaf path in such trees. More importantly, they learn about deriving and solving equations and analyzing the asymptotic behavior of functions. We do this by engaging the students in the following game between Alice and Bob. The game is based on the following problem from [2]: You are doing some stress-testing on various models of glass jars to determine the height from which they can be dropped and still not break. The setup for this experiment, on a particular type of jar, is as follows. You have a ladder with n rungs, and you want to find the highest rung from which you can drop a copy of the jar and not have it break. We call this the highest safe rung. You have a fixed "budget" of $k \geq 1$ jars. By $\mathit{HSR}(n, k) = q$ we summarize Alice' claim that she can determine the highest safe rung for a ladder on n rungs and a budget of k jars with at most q experiments. The parameters to the refutation protocol are: X is the set of sequences of questions to Bob of the form $\mathit{breaks}(j)$: Does the jar break at rung j? Y is the set of sequences of answers from Bob of the form true or false. The answers must be consistent: if $\mathit{breaks}(j_1)$ is true then $\mathit{breaks}(j_2)$ holds for all $j_2 > j_1$ and if $\mathit{breaks}(j_1)$ is false then $\mathit{breaks}(j_2)$ does not hold for all $j_2 < j_1$. The refutation function for $\mathit{HSR}(n, k) = q$ is $\mathit{refute}(n, k, q, x, y)$ returns true if (y is not consistent with x) or (y contains more than k true answers) or (x contains more than q questions) or the highest safe rung is not uniquely determined.

We give the following claims to choose from: $\mathit{HSR}(9, 2) = 6, \mathit{HSR}(9, 2) = 5, \mathit{HSR}(9, 2) = 4, \mathit{HSR}(9, 2) = 3$. Alice chooses claim $\mathit{HSR}(9, 2) = 4$ after finding a decision tree whose longest root to leaf path is 4. Bob also tries to find a decision

tree of depth 4 but he cannot. Therefore Bob decides to refute Alice' claim. However, Alice successfully defends her claim. Bob tries to improve his decision tree construction based on the input he got from Alice through the game and he now also succeeds to find a decision tree for $HSR(9,2) = 4$.

Maximum Independent Set An independent set in a graph is a set of mutually nonadjacent vertices. The problem of finding a maximum independent set in a graph, IndSet, is one of most fundamental combinatorial NP-hard problems. The refutation protocol is: Alice constructs a graph G with at most n nodes and deposits her secret independent set I_1. Alice gives G as well as the size of I_1 to Bob. In other words, I_1 is a secret for now. Bob has 10 minutes to construct his independent set I_2 which he gives to Alice and she checks it. Alice reveals her secret set I_1 and Bob checks it. Bob wins iff $size(I_2) \geq size(I_1) * 0.9$. A few of the topics that students learn while playing the game: The difference between checking and searching algorithms; How to hide secrets; How to search for secrets; Approximation Algorithms.

Min-Max Calculus is a constructive domain that can be covered with the game using modern tools like WolframAlpha. We choose claims of the form: $\forall x \in X \exists y \in Y : f(x,y) \geq c$. The claim is of the form $\forall\exists$, therefore the refutation protocol is (Alice claims, Bob tries to refute): Bob chooses x_0. Alice chooses y_0. Bob wins iff $f(x_0, y_0) < c$.

Example: Claim $GoldenRatio(c)$: $min_{x\in[0,1]} max_{y\in[0,1]}(x*y+(1-x)(1-y^2)) \geq c$, where c in $[0,1]$. If the players are perfect, claim $GoldenRatio(0.62)$ is refutable while claim $GoldenRatio(0.61)$ is not. To find out why, calculus knowledge is needed.

For mathematical claims like this, it makes sense to close them under negation and also allow:
Claim $NegGoldenRatio(c)$: $\exists x \in X \forall y \in Y : f(x,y) < c$. When Bob refutes claim $GoldenRatio(c)$ successfully, he claims a new claim $NegGoldenRatio(c)$ to find out whether his refutation was accidental.

Renaissance Games Consider yourself moved back a few hundred years to the time of the Renaissance in Italy. The mathematicians challenged each other regarding the best algorithms to solve problems. Tartaglia played the role of Alice and Fior the role of Bob in a game about cubic equations. Tartaglia won because he had discovered a technique to solve Fior's equations while Fior could not solve the ones posed by Tartaglia.

They used the following adapted refutation protocol: x and y were delivered simultaneously, i.e., y does not depend on x. Alice solves Bob's equations and Bob solves Alice' equations. $X = Y$ = sets of cubic equations. x = thirty cubic equations from Alice. y = thirty cubic equations from Bob. $refute(x,y) =$ Alice solves fewer equations in y from Bob than Bob solves in x from Alice within two hours.

Note that those renaissance claims are not mathematical claims in the sense of the Highest Safe Rung claims or the Min-Max claims. Tartaglia could not be 100% sure that he would win. But he thought that Fior had not discovered the solution technique that he had and this gave Tartaglia a significant edge. But the game is interesting and the winner demonstrates that he/she has better algorithms.

3 The Specker Challenge Game

The Specker Challenge Game (SCG) is a *mechanism* for encouraging a number of players to engage in the refutation protocol described above. The game revolves around one central pool of claims. Players are encouraged to add claims to the pool by either proposing new claims or strengthening existing claims. Players are also encouraged to test claims already existing in the pool through the refutation protocol described above. Players are evaluated based on their contribution to the pool of claims. A contribution is either a non-refuted claim (i.e., a claim that has been successfully defended) or a refutation of existing claims.

3.1 Game Benefits

The game motivates players to participate in a well defined dialog about a specific domain. This is important for educational purposes where students learn about the domain through the dialog. Stirring up the communication between geographically dispersed students in online education environments is a desirable feature. Players are engaged because they will either win the game or win constructive feedback. If Alice loses, she can study the y she got from Bob that made her lose. Maybe the y has a structure that gives a hint at Bob's clever technique for producing y. If Bob loses, he can study the x that Alice created.

Also, the structured, domain-centered communication between scholars helps stirring up the innovation in a specific domain.

A second benefit of the game is the accumulation of knowledge in a structured manner. The central pool of undisputed claims is also a useful byproduct of the game. It represents the state of the art knowledge about the domain. Note that players, scholars in this case, can still keep important pieces of information to themselves.

Consider claims of the following algorithmic form: Alice claims she has an algorithm that solves any problem x in X with a solution $y(x)$ in $Y(x)$ that has quality $q(Y(x))$. The refutation protocol is: Bob constructs a hard problem instance x_0 in X and Alice must find a solution y of the claimed quality. If she cannot, Bob wins. Note that Alice can keep her algorithm for solving problems in domain X secret and Bob can keep his algorithm for generating hard problems in domain X secret. Yet a central pool of hard problems and some good solutions for them is populated. Games of this kind drive the scholars towards a common good, although they are egoistic: to find the best algorithm and to pose the hardest problems. The scholars create social welfare through their egoistic behavior thanks to the mechanism design the game uses.

3.2 Game Versions

Intensional Vs. Extensional Set of Claims Before playing the game all three parties need to agree on a how claims are expressed. A domain expert is expected to provide such specification. In the intentional version, the expert defines a language to express a usually huge set of claims that may be proposed. In the extensional version, the expert enumerates the universe of all possible claims that may be proposed.

Board Vs. Avatar SCG/Board is the informal version of the game where two students challenge each other and together check the rules of the game. SCG/Board Extensional is a convenient combination because of its low overhead. The students can control their educational experience by choosing claims in a domain they want to explore further. The universe of possible claims is usually enumerated by the instructor.

The main attraction of SCG/Avatar is that players get the opportunity to create their "organism" that needs to fend for itself in a real virtual world inhabited by organisms created by other players.

This version of the game works well when the domain is sufficiently simple so that the task of proposing and opposing can be easily programmed. SCG/Avatar brings to the game a strong software development component and a very fun competitive/collaborative environment. The avatars live somewhere on the web, written in your programming language of choice, and compete with each other in a full round-robin or Swiss style tournament. We have observed that students during the running game changed their avatar to improve its performance against the avatars of their peers. When they come to class, they get a simple baby avatar that can walk and talk (it can briefly survive on the web) but does not have much intelligence. Then they add intelligence that is hopefully better than the intelligence that their peers add. The game creates a ranking of the avatars that gives feedback to the students about their algorithm and software development skills.

The Highest Safe Rung and Min-Max games could be easily played using SCG/Avatar by teaching the avatars enough calculus so that they propose strong claims and successfully refute weak claims. We used a more complex version version of SCG/Avatar based on the Constraint Satisfaction Domain.

3.3 The Scientific Community Game Mechanism

SCG is also an acronym for Scientific Community Game in which scholars compete for reputation gained by proposing hard to refute claims or refuting existing claims. The game proceeds according to the following rules:

- The game is a two player game.
- The game consists of a fixed number of turns. A turn can only take a fixed amount of time.
- Scholars start with a fixed and equal amount of reputation.
- The sum of reputations remains constant throughout the game.
- There is only one central pool of claims. Initially, the pool is empty.
- Claims contain a real numbered field in the range $[0..1]$ referred to as strength. Claims also contain another real numbered field in the range $[0..1]$ referred to as confidence.
- The credibility of a claim is the result of multiplying their confidence field by the reputation of their proposing scholar at the time the claim is proposed.
- In each turn except the final turn, every scholar with a positive reputation must either:
  - propose at least one new claim to be added to the central pool of claims.
  - strengthen at least one claim from the central pool. Strengthening is not possible in the first turn and is not allowed in the final turn.
- Scholars can also refute existing claims.

– Scholars strengthen claims by increasing their strength fields from s_i to s_t. When they do so, they gain $(s_t - s_i)$ multiplied by the credibility of the claim.
– Scholars refute claims by engaging in an instance of the refutation protocol mentioned above. The result is a recognition factor $[-1..1]$. The opposed scholar reputation gain is the recognition factor multiplied by the credibility of the claim.

Below, we give an explanation/rationale for some of the less obvious game rules:

It is important to keep the game a two-player game so that a player can regain its reputation lost when its claims are strengthened by opponents. In a multi-player version, players that come after weak players have the advantage of being able to gain reputation by strengthening and refuting claims made by those weak players. Furthermore, in a multi-player version, players can form coalitions in which one player loses reputation for another player. These kind of coalitions are non-productive from the point of view of the game purpose. The two player requirement is not a limitation though, as a multi-player version can be derived from this two player version via a tournament made up of multiple rounds (such as the Swiss-style tournament used in the FIFA world cups).

Confidence is a real number in $[0, 1]$ that indicates how confident the proposer is in the claim. The confidence should reflect the amount of effort that has been put into the claim. Confidence 1 means that we believe that the claim cannot be opposed (i.e. strengthened or refuted). For example, one could use confidence 1 for a claim for which the scholar has a proof.

Strength is a real number in $[0, 1]$ that indicates the strength of the claim. For example, in the min-max domain mentioned earlier, the claim $\forall x \in X \exists y \in Y : f(x, y) \geq 0.62$ is stronger than the claim $\forall x \in X \exists y \in Y : f(x, y) \geq 0.61$. In some domains, there is no inherent notion of the strength of the claim, or there is an inherent notion of strength that cannot be expressed by a real number in the range $[0, 1]$. The O-notation domain is an example of such domains. In this case, strength is set to 1 and it is no longer possible to strengthen claims.

Claims also have a credibility that is proportional to both it's proposing scholar's "confidence" in the claim and it's proposing scholar's reputation at the time of proposing the claim.

Since the sum of reputations is preserved and the initial reputation sum is positive, we are guaranteed to have at least one scholar with a positive reputation. Scholars with positive reputation are required to be active so that the game is guaranteed to produce an accumulation of claims.

Scholars with positive reputation stay active by either proposing at least one new claim or strengthening at least one existing claim. Scholars can also try to refute existing claims for a possible reputation gain that will help them win the game. Strengthening a claim is a sort of opposing an existing claim and proposing a new (stronger) claim at the same time. That's why strengthening qualifies both as a proposition and as an opposition at the same time.

It is meaningless to propose new claims at the final turn as the players do not gain reputation by proposing new claims they get reputation by successfully supporting their proposed claims. Since there are no more turns for their opponent to try to refute their claims, it becomes meaningless to propose new claims in the final turn.

Strengthening is not allowed in the final turn as it makes it possible for a player to get away with reputation gained by over-strengthening an existing claim. It is also essential for scholars to get the chance to refute the strengthening of their claims made by opponents as they lose reputation when their claims are strengthened by other scholars. This also enhances the quality of the final set of claims in the central pool as they are less likely to contain overly strong claims.

Scholars with negative reputation cannot propose any new claims or strengthen any existing claims, as their new claims would have negative credibility leading to opponents losing reputation when they strengthen or successfully refute these claims. Scholars with negative reputation can only try to refute existing claims in order to increase their reputation.

The refuting protocol recognizes claims by a recognition factor in $[-1, 1]$ A recognition factor of 1 means that the other scholar has completely failed to refute that claim. A recognition factor of -1 means that the other scholar has completely succeeded to refute that claim.

Scholar's final reputation is the sum of the scholar's initial reputation plus the reputation gains and losses; thus reflecting the past performance of the scholar. The scholar with the highest final reputation wins the game.

There is an issue with the axioms as they are currently formulated: When a scholar with low reputation refutes the claim of a scholar with high reputation, the low reputation scholar will significantly increase its reputation. But when the low reputation scholar loses, it will also lose a high amount of reputation. This might discourage scholars with low reputation from "testing" claims made by scholars with high reputation.

3.4 Equilibrium

SCG has a natural equilibrium state. When all players are perfect, they propose claims that can neither be refuted nor strengthened. The other players will recognize those claims as "optimal". Everybody will keep their initial reputation and the game is a tie. In real games with imperfect players, there is a lot of refuting and strengthening happening.

4 Applications

4.1 Education

The Specker Challenge Game works very well to create a productive interaction between students of constructive domains, like STEM fields. The Specker Challenge Game focuses the student interaction on a specific domain and gives the students valuable hands-on experience when they propose and oppose.

Software Development We have used SCG/Avatar successfully in software development courses. Implementations of $SCG(CSP)$ have been used at Northeastern University to teach several semesters of a senior level course on software development [4, 3]. Throughout the course students learn not only about the rigors of software organization

and evolution, but about some details of SAT/CSP algorithms. The feedback from both course staff and regular competitions keeps students involved, putting their best ideas into the avatars.

The course invites students to master a wide variety of CS topics ranging from Theory of Computation, Discrete Structures (combinations, permutations, simple combinatorics), Calculus (maximizing and minimizing multi-variable rational functions) to Object-Oriented design and Software Engineering.

An important part of teaching is to evaluate whether the students understand the computational processes and whether they can effectively use them in practice. A round-robin contest is the perfect tool to give the students quick feedback how well they do compared to the other students. Furthermore, the agents help each other with testing.

When students come to the course, they get a baby agent that must be equipped with intelligence to win. We got reports in Spring 2009 that students were so immersed into the game that they ignored other courses and put all their effort into winning. We used the rule that after two weeks of contests, the source code had to be revealed. This on one hand lead to rapid improvements to the agents but it also contributed to the competitive nature of the game.

We plan to further develop our use of the Specker Challenge Game for teaching Software Development. There is significant value in giving the students a baby agent that they have to nurture into an "intelligent adult" using state-of-the-art software development technology.

Algorithms We have also used SCG successfully in two undergraduate algorithms courses [6]. Here we used the "board" version of the game (SCG/Board) where the two participating students jointly check the game rules. We use the game to help the students to create ideas about how to solve textbook problems. We see value in having students defend claims that are true and refute claims that are false. A successful defense of a claim maybe a baby step towards a proof of the claim.

4.2 Innovation

We believe that the small innovation steps we have observed in undergraduate competitions can be achieved on a larger scale when world-class researchers participate (e.g., PhD students in algorithms) both for SCG/Avatar and SCG/Board.

Optimization Problems How can we drive innovation towards better algorithms for NP optimization problems? NPO is the class of all NP optimization problems. The ingredients to an optimization problem are the set of input instances or problems, the set of feasible solutions, and the measure defined for a feasible solution. In a nutshell, NPO problems are optimization problems whose feasible solutions are short and easy to recognize.

For optimization problems, the game proceeds as follows. Alice claims that she has an algorithm to solve a family of optimization problems within a given resource bound and a given solution quality. The refutation protocol: Bob gives Alice a problem (x) in the family and if Alice cannot produce a solution (y) within the given resource bound

or the promised quality, Alice loses. Solution quality and resource consumption usually manifest in the game in the form of claim strength.

Using SCG/Avatar, we design a virtual scientific community that through the avatar game improves its knowledge. The game is designed in such a way that the winning avatars have the best approximation algorithms for the NPO problems under discussion.

SCG/Avatar offers a new way to evaluate algorithms without resorting to an asymptotic analysis. Good asymptotic behavior is important but games are won also by clever implementations that use small constant factors. The problem instances allowed to appear in the game are all bounded by a large constant.

4.3 Better Software Development Process for Computational Problems

Developing software for computational problems is an important ongoing activity. The claims are of the form: $Claim(X, Q, R)$: I have a program that solves instances in X with quality Q and resource bound R. SCG/Avatar provides a new software development process for such software that is based on both collaboration and competition. For example, instead of having 4 people working as one team, we use 2 people to develop Alice and 2 to develop Bob and the competitions between Alice and Bob will improve them. The process works better when more avatars are involved.

The benefits of the process are:

- Engages Software Developers. Nobody wants to see their avatar suffer and being beaten by their peers. The competitions typically run over several hours and we saw software developers stay up at night and improve their avatar between rounds in the same competition.
- Meaningful Measurement. The tournament ranking provides an objective evaluation of the avatars and the teams behind them with respect to software development and algorithm skills. The game is designed in such a way that the winning avatar must demonstrate better algorithms and their correct implementation.
- Encourages Good Software Engineering Properties. There is a lot of pressure between the competitions to update the avatar's code quickly. This can only be done if the code has good software engineering properties such as modularity.
- Encourages Good Algorithms. Winning avatars often have superior algorithmic knowledge that they exploit in the game.
- Testing. The avatars are thoroughly tested through the opposing activity.

4.4 Hiring

Companies can benefit from SCG/Avatar by screening potential employees with the game. When a company seeks algorithmic/implementation skills in a given algorithmic problem solving domain, they define a game for this domain and let potential employees participate. The potential employees might create useful knowledge for which they are compensated. For example, the potential employee competition might create an algorithm that beats the algorithm that was created in-house.

5 Evaluation of the Specker Challenge Game/ Disadvantages

The Specker Challenge Game, because its very flexible refutation protocol, covers a broad area. Because of this coverage, the effort of learning the game can be quickly amortized. But one disadvantage of the game is that students and software developers need to learn the game mechanics of a scientific community.

The Avatar version requires a very well functioning game engine to work well. The administrator needs to painstakingly check all avatars whether they follow the rules. The Avatar version requires a software tool to quickly generate baby avatars. We hope to make this parameterized engine freely available but we are not there yet.

We found SCG/Avatar to be addictive. Students identified with their avatars and did not want to see them suffer. We had to advise the students to balance their efforts with other courses.

6 Origins of the Specker Challenge Game

Initially, we used the domain of Boolean CSP: Boolean constraint satisfaction problems, motivated by our joint work with Ernst Specker [8]. Specifically, our problem domain X was the set of CSP *formulas*, where a formula is a sequence of constraints, each over a set of variables. For a given CSP formula $x \in X$ we use $V(x)$ to represent the set of all variables used in its constraints. Feasible solutions, $S(x)$, are assignments to a formula's variables, defined as all maps whose domain is $V(x)$, and range is boolean values, $\{true, false\}$.

For CSP, the easiest way to define a sub-domain for a claim (*i.e.*, X') is to do so intensionally by giving a set of *relations* that can be used to create legal constraints. The function $fsat(x, s)$, which calculates the fraction of satisfied constraints in x using assignment s, is used for predicate descriptions.

A scholar might offer a claim using the relation 1-IN-3, which is *true* when exactly one of its three variables is assigned *true*. The claim proposed could be: Alice claims: There exists a CSP formula using only the 1-IN-3 relation so that for all assignments to the formula at most the fraction 0.4 will be satisfied.

$$\langle\langle\{\text{1-IN-3}\},\, Q\rangle,\, 0.4\rangle \quad \text{with} \quad Q(x,\, s) = (fsat(x,\, s) \leq 0.4)$$

If refuted by Bob, the offering scholar Alice would need to provide a CSP instance using the 1-IN-3 relation, *e.g.*:

$$x \;\equiv\; \langle\, (\text{1-IN-3}\ \{a,\, b,\, c\}),\ (\text{1-IN-3}\ \{a,\, b,\, d\}),\ (\text{1-IN-3}\ \{b,\, c,\, d\})\,\rangle$$

We can quickly see that with the defined predicate, Q, it is rather easy to refute the claim given this problem instance. Though not all assignments provide a good fraction of satisfied constraints (*e.g.*, the all *false* assignment does not satisfy any), there are multiple assignments that do. Two possible solutions and their satisfied fractions are:

$$\begin{aligned} fsat(x,\, \{\, a \mapsto false,\, b \mapsto true,\, c \mapsto false,\, d \mapsto false\,\}) \;&=\; 1 \quad \text{and} \\ fsat(x,\, \{\, a \mapsto true,\, b \mapsto false,\, c \mapsto false,\, d \mapsto false\,\}) \;&=\; 2/3 \end{aligned}$$

For formulas made only of this relation, 1-IN-3, there always exists an assignment that will satisfy at least $4/9$ of all constraints. This is known as a *P-optimal* threshold for this specific CSP problem [7]: we can satisfy this fraction quickly (in polynomial time), but to do better than $4/9 + \epsilon$ for $\epsilon > 0$ makes the problem NP-hard. We get this value by maximizing the polynomial: $3 \cdot p \cdot (1-p)^2$, which we derive from the truth table of 1-IN-3, eventually obtaining the maximum value at $p = 1/3$.

7 Related Work

TopCoder [11] has connections to SCG/Avatar. TopCoder offers algorithm competitions where you need to defend your own program and try to find bugs in programs of others. But there are big differences: Specker Challenge Game has claims which can be flexibly defined while in TopCoder they have a fixed form: I have an algorithm that solves the problem correctly. If you want to refute my claim you must provide a test-input where the code fails. Also in SCG/Avatar the avatars must create claims which is absent in TopCoder.

The Specker Challenge Game is patterned after a real scientific community: This related work is quite old. Scientists are encouraged to offer results that are not easily improved. They must offer results that they can successfully support. They strengthen results, if possible. They must stay active and publish new results or oppose current results. They want to become famous! How to turn predicate calculus statements with an even number of alternating quantifiers into a game is well-known folklore.

[9] provides excellent background information on SCG. However, the idea of defining games with claims/refutation protocols is not touched. The chapter on Evolutionary Game Theory (chapter 29) is most relevant to SCG, although traditional evolutionary game theory works with infinitely many players. In this world, the players are organisms that meet other organisms from which they buy products and to which they offer products. The 2 organisms play a fixed, 2 player game.

[5] gives further information on the Specker Challenge Game. [10] provides a good introduction to game design. [1] promotes the use of serious games in computer science programs but from a different angle than this paper.

Acknowledgements

We would like to thank Novartis/NIBR for partially supporting the development of the Specker Challenge Game. Alex Dubreuil was the teaching assistant for several courses where the Specker Challenge Game was used and he gave us valuable feedback and encouragement. We would like to thank Pete Manolios for his feedback on the game and to Shriram Krishnamurthi for the Renaissance Game connection.

References

1. K. Becker and J. R. Parker. "Serious Games + Computer Science = Serious CS". *J. Comput. Small Coll.*, 23:40–46, December 2007.

2. J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.
3. K. Lieberherr. Software Development: CSU 670 Fall 2009. Website, 2009. http://www.ccs.neu.edu/home/lieber/courses/cs4500/f09/cs4500-f09.html.
4. K. Lieberherr. Software Development: CSU 670 Spring 2009. Website, 2009. http://www.ccs.neu.edu/~lieber/courses/csu670/sp09/csu670-sp09.html.
5. K. Lieberherr. The Specker Challenge Game. Website, 2009. http://www.ccs.neu.edu/~lieber/evergreen/specker/scg-home.html.
6. K. Lieberherr. Algorithms and Data CS 4800, Fall 2010. Website, 2010. http://www.ccs.neu.edu/home/lieber/courses/algorithms/cs4800/f10/course-description.html.
7. K. J. Lieberherr. Algorithmic extremal problems in combinatorial optimization. *Journal of Algorithms*, 3(3):225–244, 1982. http://www.ccs.neu.edu/~lieber/p-optimal/karl-algo-extremal.pdf .
8. K. J. Lieberherr and E. Specker. Complexity of partial satisfaction. *Journal of the ACM*, 28(2):411–421, 1981. http://www.ccs.neu.edu/~lieber/p-optimal/JACM1981.pdf .
9. N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.
10. J. Schell. *The Art of Game Design: A Book of Lenses; electronic version*. Elsevier, Burlington, MA, 2008.
11. TopCoder. The TopCoder Community. Website, 2009. http://www.topcoder.com/.