Crowdsourcing Formal Decision Making Using Generalized Semantic Games

Ahmed Abdelmeged

We organize formal scientific knowledge in an objectively disputable form and make formal science available to a wider audience.

My Thesis

Semantic games of interpreted logic statements provide a useful foundation for building successful crowdsourcing systems for deciding formal science claims.

Applications

- Formal science Wikipedia.
- Solving computational problems.
- Solving hardware and software verific problems.
- Education in formal science.

Wikipeida has a subjective process for disputing claims,

can make formal science claims about computational problems and h/w and s/w

Students get feedback on the position they take on formal science claims. With minimal instructor involvement.

Outline

- Introduction
- Related Work
- Proposed Approach
- Evaluation
- History and Future Work

Deciding Formal Science Claims

- A Formal Science Claim Family $\langle \varphi(p), A \rangle$ is a parameterized logical formula, interpreted in a "rich", computable structure A.
- $S(c \in [0,2]) = \forall x \in [0,1]: \exists\, y \in [0,1]: x + y > c$
  - The structure consists of the natural numbers with +, >, ...

Example Formal Science Claim Protein Folding (1)

- Proteins are made of long chains of amino acids (~100's).
- Some amino acids attract and repulse, some amino acids are hydrophilic and some are hydrophobic.
- These forces determine the native state, the most stable 3D-structure (a.k.a. folding) of a protein.

Example Formal Science Claim Protein Folding (2)

- nativeState(p ∈ Proteins, f ∈ Foldings(p)) :=
  ∀ f2 ∈ Foldings(p) : energy(p, f) ≤ energy(p, f2)
- hasNativeState(p ∈ Proteins) :=
  ∃ f ∈ Foldings(p) : nativeState(p, f)
  - The logical formula is intended to describe the input to be provided by humans.
    - Supported by the "rich" structures, implemented in Turing complete programming language.
  - For most claim families, there is no known (efficient) decision procedure. Humans are needed to provide ***justified decisions***.
  - FoldIt!

"Predicting protein structures with a multiplayer online game" --Seth Cooper, et al., 2010

Outline

- Introduction
- Related Work
- Proposed Approach
- Evaluation
- History and Future Work

Current Approaches to Deciding Formal Science Claims

- Proofs.
  - Too challenging for the crowd.
- Model checkers.
  - Don't handle "rich" structures.
- Semantic Games.

Decision Making Using Semantic Games (SGs)

- A semantic game for a given claim $\langle \varphi(p_0), A \rangle$ is a game played by a verifier and a falsifier, denoted SG($\langle \varphi(p_0), A \rangle$, verifier, falsifier), such that:

- A |= $\varphi(p_0)$ <=> the verifier has a winning strategy.

Toy Example

- $S(c \in [0,2]) = \forall x \in [0,1]: \exists\, y \in [0,1]: x + y > c$
- S(c) is true for $c \in [0,1)$ and false for $c \in [1,2]$
- Best strategy:
  - for the falsifier: x=0
  - for the verifier: y=1

Toy Example: SG Trace

SG(∀x ∈ [0,1]: ∃ y ∈ [0,1]: x + y > 1.5 ,  , )

 Provides 1 for x | Weakening (too much!)

SG(∃ y ∈ [0,1]: 1 + y > 1.5 ,  , )

 Provides 1 for y | Strengthening

SG(1 + 1 > 1.5 ,  , ) Wins

```
\forall x \in [0,1] \exists y
\in [0,1]: x \cdot y + (1-x)
\cdot (1-y^2) \geq 0.62

\exists y \in [0,1]: 0.5 \cdot
y + 0.5 \cdot (1-y^2) \geq
```

Moves of SG(⟨φ, A⟩, v, f)

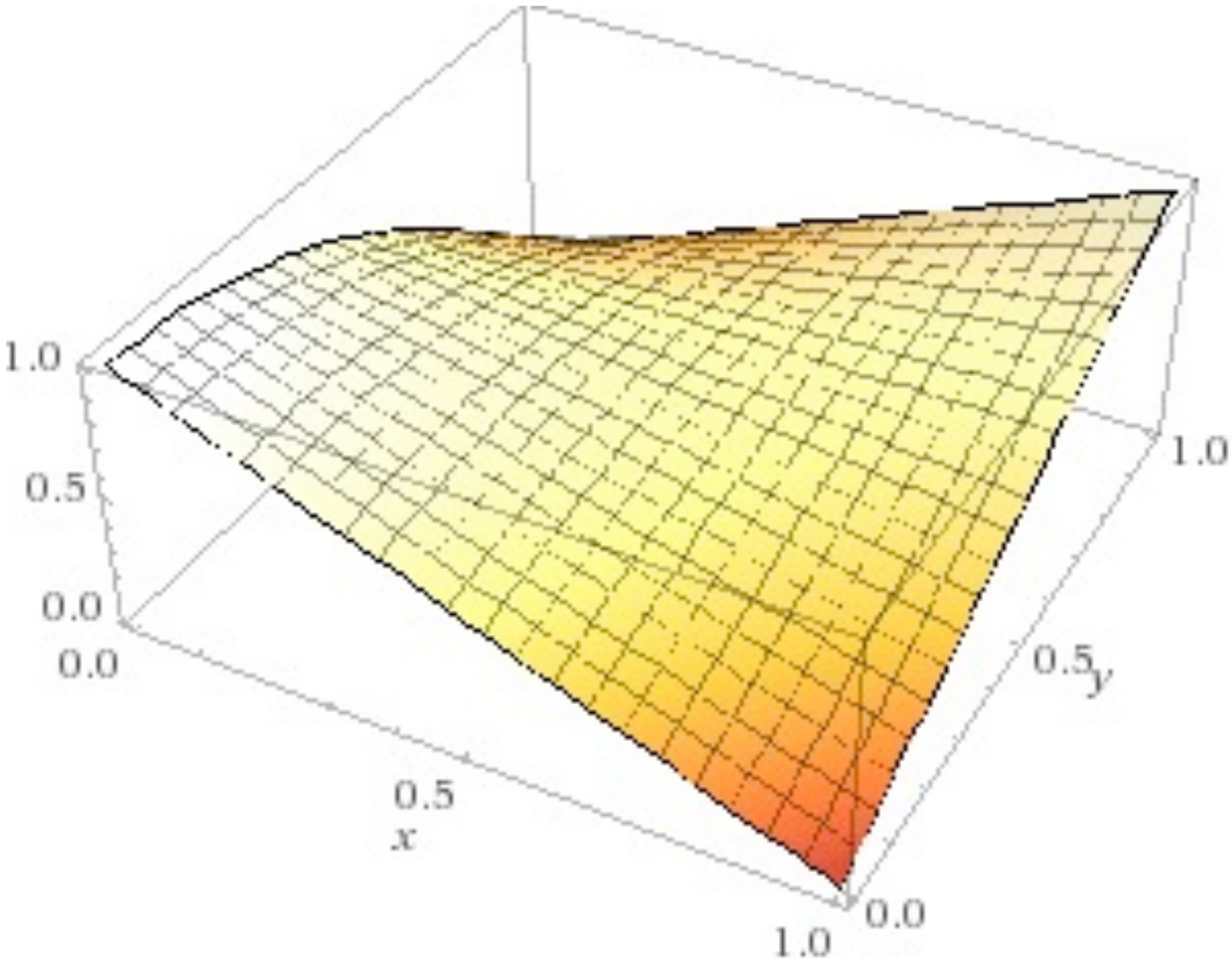
φ	Move	Next Game
$\forall x : \psi(x)$	f provides x_0	SG(⟨$\psi[x_0/x]$, A⟩, v, f)
$\Psi \wedge \chi$	f chooses $\theta \in \{\psi, \chi\}$	SG(⟨θ, A⟩, v, f)
$\exists x : \psi(x)$	v provides x_0	SG(⟨$\psi[x_0/x]$, A⟩, v, f)
$\psi \vee \chi$	v chooses $\theta \in \{\psi, \chi\}$	SG(⟨θ, A⟩, v, f)
$\neg \psi$	N/A	SG(⟨ψ, A⟩, f, v)
$P(t_0)$	v wins if $P(t_0)$ holds, o/w f wins	

"The Game of Language: Studies in Game-Theoretical Semantics and Its Applications"
-- Kulas and Hintikka, 1983

Strategies

- A strategy is a set of functions, one for each potential move.

$$\mathrm{x} \cdot y + (1 - x) \cdot (1 - y^2)$$



16

Example

- For $SP(c) := \forall x \in [0,1] \exists y \in [0,1] : x \cdot y + (1-x) \cdot (1-y^2) \geq c$
  - A potential falsifier strategy is: provideX (c){ 0.5 }.
  - A potential verifier strategy is: provideY (x, c){ x }.

Example: SG Trace

SG($\forall x \in [0,1] \exists y \in [0,1] : x \cdot y + (1-x) \cdot (1-y^2) \geq 0.62$, ✔ , ✖)

✖ Provides 0.5 for x | Weakening (too much!)

SG($\exists y \in [0,1] : 0.5 \cdot y + 0.5 \cdot (1-y^2) \geq 0.62$, ✔ , ✖)

✔ Provides 0.5 for y | Strengthening

SG($0.625 \geq 0.62$, ✔ , ✖) ✔ Wins

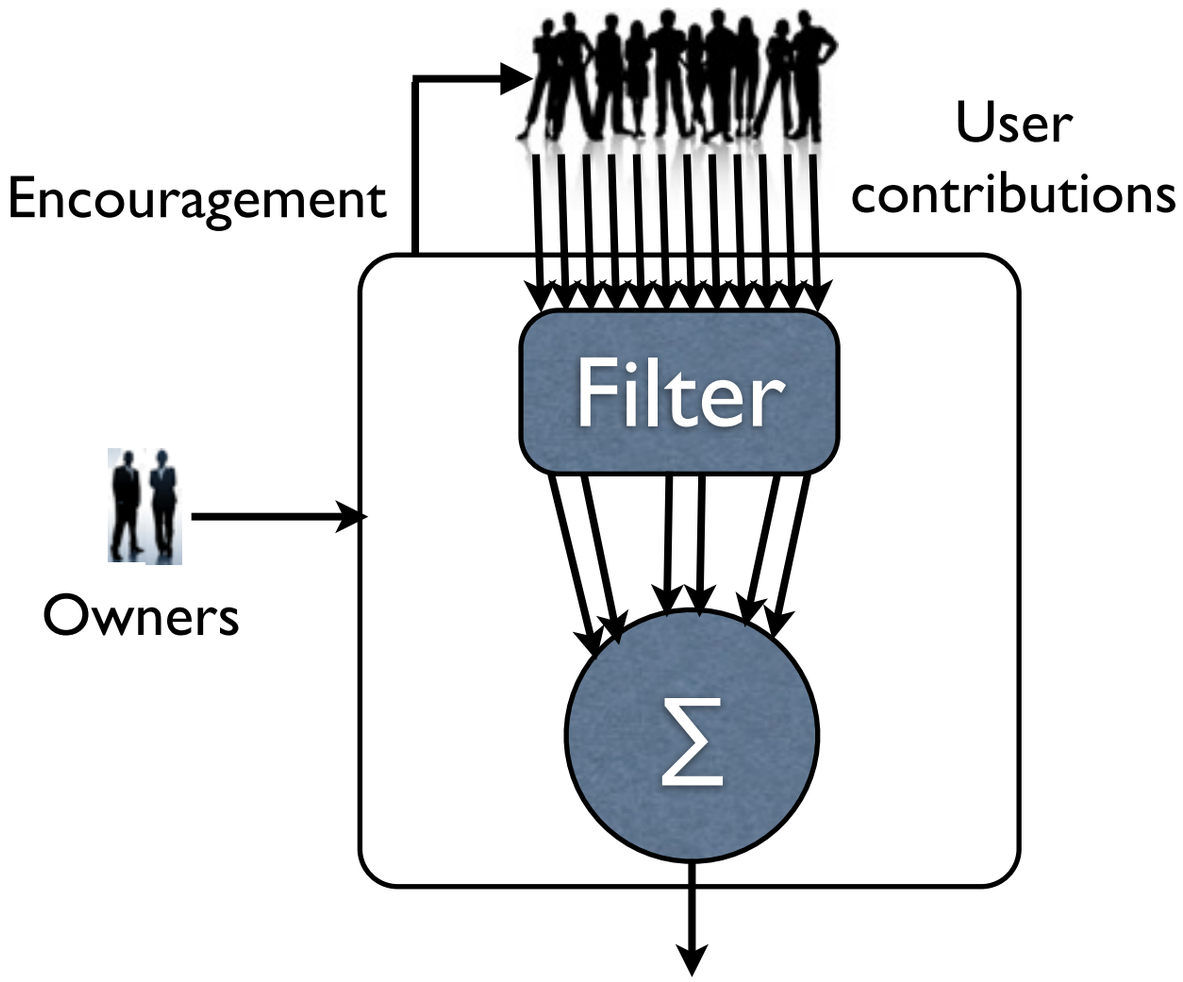
```
\forall x \in [0,1] \exists y
\in [0,1]: x \cdot y + (1-x)
\cdot (1-y^2) \geq 0.62

\exists y \in [0,1]: 0.5 \cdot
y + 0.5 \cdot (1-y^2) \geq
```

SG Properties
(Relevant to our approach)

- SG winners drive their opponents into contradiction.
- Faulty verifier (falsifier) actions can produce a false (true) claim from a true (false) one.
- Faulty actions will be exposed by a perfect opponent leading to a loss.
- Winning against a perfect verifier (falsifier) implies that the claim is false (true).
- Losing an SG implies that either you did a faulty action or you were on the wrong position.

C/S Systems



Challenges

- Define user contributions.
- Evaluate users and their contributions
- Combine user contributions.
- Encourage and retain users.

"Crowdsourcing systems on the world-wide web"

--Anhai Doan, Raghu Ramakrishnan and Alon Y. Halevy, 2011

Example C/S Systems (1)

- Informally specified tasks:
  - Simple: Image labeling (ESP Game) and web page classifiers (Ipeirotis et al.).
    - Combined through majority voting.
  - Complex: Crowdforge (Smus et al.) and Wikipedia.
    - Combined through manual effort.

Example C/S Systems (2)

- Formally specified tasks:
  - FoldIt! (Cooper et al.), EteRNA(Treuille et al.), PipeJam(Ernst et al.) and Algorithm development competitions at TopCoder.
- We provide a general, collaborative, framework.

Outline

- Introduction
- Related Work
- Proposed Approach
- Evaluation
- History and Future Work

Overview

- We use SGs to collect evidences of truth of claims an skill/strength of users.
- Egoistic users produce social welfare.

SGs and C/S Systems

- SGs provides a foundation to:
  - Combine user contributions : winner's position and moves are *assumed* to be correct.
  - Evaluate users: winner is *assumed* to be more skilled.
- SGs can help retaining users as they can be fun to play and watch.
- SGs have a collaborative nature. Winners provide information to losers. SGs help "educate" the crowd.

Proposed Approach

- Owners provide a claim c, the unreliable users in the crowd provide strategies (a.k.a. avatars) for playing SG(c,-,-).
- We get the avatars to play numerous SGs. Then we combine their outcome to:
  - Estimate the truth likelihood of c.
  - Estimate the strength of avatars.
- Users update their avatars and then we iterate.

First Shot: Using SGs

- Given a claim c, run numerous SG(c, v, f) where v, f are chosen at random from the crowd.
  - The more often the verifiers win, the more likely c is true.
    - Users with more wins have better strategies.
- Suppose c is true (false), and the falsifier (verifier) wins. This reduces the estimated truth likelihood of c.
- Suppose c is true (false), and the falsifier (verifier) loses. This reduces the estimated skill level of the falsifier (verifier).

Generalizing SGs

- First ask users for their favorite position.
- If both choose the same position, force one to play the devil's advocate.

Winner	Forced	Payoff (u, !u)	Truth evidence
u	None	(1, 0)	Pos(u)
u	u	(1, 0)	None
u	!u	(0, 0)	Pos(u)

Estimating Claim Truth Likelihood

- Truth Likelihood = E_v / (E_v + E_f), where E_v (E_f) is the number of times the non-forced verifier (falsifier) wins. [UNW]
- Each win is weighted by the strength of the opponent. [W8D]

Estimating User Skill: Simple Approach

$$Wins_{SM}(U_i) = \sum_j Payoff(U_i, U_j)$$

$$Losses_{SM}(U_i) = \sum_j Payoff(U_j, U_i)$$

$$Str_{SM}(U_i) = Wins_{SM}(U_i)/(Wins_{SM}(U_i) + Losses_{SM}(U_i))$$

- The fraction of wins against a non-forced players [SM]

```
W\!ins_{SM}(U_i) &=& \sum_{j} P\!ayo\!f\!\!f\!\,({U_i},{U_j}) \\
Losses_{SM}(U_i) &=& \sum_{j} P\!ayo\!f\!\!f\!\,({U_j},{U_i}) \\
Str_{SM}(U_i) &=& W\!ins_{SM}(U_i)/(W\!ins_{SM}(U_i)+Losses_{SM}
```

Estimating User Skill: Iterative Approach

$$
\begin{aligned}
Str^{0}_{IT}(U_i) &= Str_{SM}(U_i) \\
Wins^{(k)}_{IT}(U_i) &= \sum_j Payoff(U_i, U_j) * Str^{(k-1)}_{IT}(U_j) \\
Losses^{(k)}_{IT}(U_i) &= \sum_j Payoff(U_j, U_i) * (1 - Str^{(k-1)}_{IT}(U_j)) \\
Total^{(k)}_{IT}(U_i) &= Wins^{(k)}_{IT}(U_i) + Losses^{(k)}_{IT}(U_i) \\
Str^{(k)}_{IT}(U_i) &= \begin{cases} 0.5, \text{if } Total^{(k)}_{IT} = 0 \\ Wins^{(k)}_{IT}(U_i)/Total^{(k)}_{IT}(U_i), \text{o/w} \end{cases}
\end{aligned}
$$

- Winning against a strong user results in a large gain. Losing against a strong user results in a small hit. [IT]

```
Str_{IT}^{0}(U_i) &=& Str_{SM}(U_i) \\                    31
W\!ins_{IT}^{(k)}(U_i) &=& \sum_{j} P\!ayo\!f\!\!f\!\,({U_i},{U_j}) * Str_{IT}^{(k-1)}(U_j) \\
```

The Crowd Interaction Mechanism (CIM)

- SGs are binary interaction mechanisms that need to be scaled to the crowd.
- CIM decides which SGs to play.
- Several tournament options.
- Should be simple and intuitive for users.
- Need a fair CIM with minimal effect on estimated user skill levels.

Sources of Unfairness

- Users u1 and u2, taking the same position on claim c, are not given the same chance if:
  - u1 and u2 play a different number of SGs against any other opponents.
  - Either u1 or u2 is forced more often.
  - There are other players that are willing to lose against either u1 or u2 on purpose.

The Contradiction Agreement Game (CAG)

- If two users choose different positions (contradiction) on a given claim. They play a regular SG.

- If two users choose the same position (agreement) on a given claim. They play two SGs where they switch playing devil's advocate.

- CAGs eliminate the forcing advantage.

A Fair CIM

- A full round robin tournament of CAGs, eliminates the potential unfairness arising from playing a different number of games against any other opponent or being forced more often.

Outline

- Introduction
- Related Work
- Proposed Approach
- Evaluation
- History and Future Work

Evaluation Approach

- We evaluate the system based on the quality of estimated truth likelihood (Et) and user strength in a set of benchmark experiments.
- Each experiment consists of:
  - A claim with a known truth.
  - A crowd of synthetic users with predetermined skill distribution.
- The quality of estimated truth likelihood is Et for true claims and (1-Et) for false claims.
- The quality of estimated user strength is the fraction of pairs of users whose rank is consistent with their predetermined skill.

Synthetic Users (1)

- A synthetic user with skill level p, denoted su_p, makes the perfectly correct action with probability p and makes the perfectly incorrect action with probability (1-p).

Example Synthetic User

$$\forall x \in [0,1] \exists y \in [0,1] : x \cdot y + (1-x) \cdot (1-y^2) \geq 0.62$$

Perfectly Correct Actions

ProvideX(c){ 0.552 }

ProvideY(c, x){ min(x, x/(2-2*x)) }

Perfectly Incorrect Actions

ProvideX(c){ 0 }

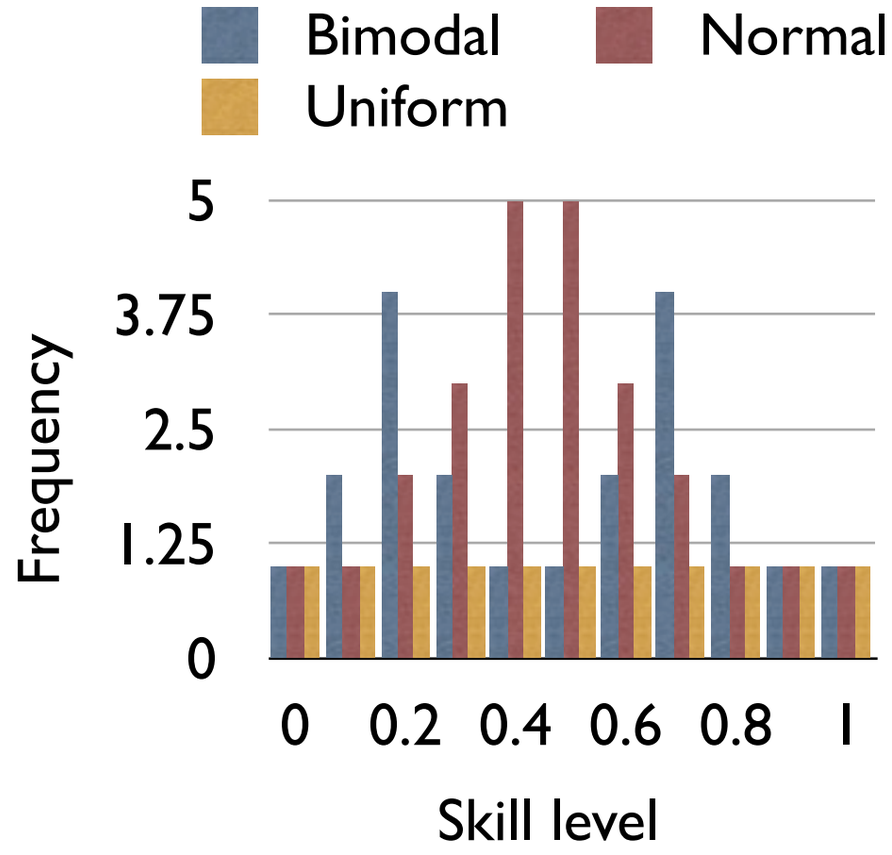
ProvideY(c, x){ x>0.5?0:1 }

Synthetic Users (2)

- A synthetic user su_p chooses its position on a claim c to be the winner position of SG (c, su_p, su_p).
- su_1 will always choose the correct position.
- Otherwise, the probability of choosing the correct position depends on the claim.

Initial Experiments

- Crowd skill distributions:
  - Normal.
  - Binomial.
  - Uniform.
- Claims:
  - SP(0.2) and SP(0.75).
- Configurations:
  - CIM : AL vs TH
  - UE : SM vs IT
  - CE : W8D vs UNW



Results

- W8D to enhances CE quality.
- Full round robin to be produce fewer inconsistent rankings than the partial round robin.
- Surprisingly, we found that the simple user evaluator to produce fewer inconsistent rankings than the iterative evaluator.

Configuration	CE quality	UE Quality
AL-SM-UNW	0.807	0.815
AL-SM-W8D	0.851	0.815
AL-IT-UNW	0.807	0.769
AL-IT-W8D	0.848	0.770
TH-SM-UNW	0.808	0.556
TH-SM-W8D	0.837	0.554
TH-IT-UNW	0.807	0.558
TH-IT-W8D	0.836	0.557

Uniform, SP(0.2)

Outline

- Introduction
- Related Work
- Proposed Approach
- Evaluation
- History and Future Work

What We Have Done (2007-2013)

- [2007-2008] Specker Derivative Game (SDG): Game of Financial Derivatives for CSP. Supported by GMO.
- [2009-2011] Specker Challenge Game (SCG): protocols instead logic sentences, propose claims, defend or refute or strengthen. Supported by Novartis.
- [2013-]Scientific Community Game (SCG): claim families defined by parameterized logic formulas, defend or refute through semantic games (instead of protocols).

"The Specker Challenge Game for Education and Innovation in Constructive Domains" -- Keynote paper at Bionetics 2010.

What We Plan To Do (Until End of August '13)

- Development & evaluation based on synthetic users.
  - Build up the benchmarks.
  - Fine tune the system.
  - Support more use cases.
  - Bring the system to the web.
- Experiment with humans writing avatars.
  - Highest safe rung problem.
  - Beat the system.

Highest Safe Rung

- Given a ladder with n rungs and k identical jars, the goal is to discover the highest rung such that the jar doesn't break when thrown from. What is the experimental plan that minimizes the total number of experiments?
- $\text{minHSR}(n \in N, k \in N) := \exists q \in N : \text{HSR}(n, k, q) \wedge \neg \text{HSR}(n, k, q-1)$
- $\text{HSR}(n \in N, k \in N, q \in N) := \exists d \in \text{DecisionTrees} : \text{HSRCorrect}(d, n, k, q)$

Potential Innovations

- Start with linear search.
- K=2.
- Reformulate: what is the maximum number of rungs that can be handled by k jars in q experiments?
- Modified Pascal Triangle.

Questionnaire

- How engaging was the experience of writing an avatar that fought on your behalf (scale from 1 to 10).
- What did you learn from your peers through the semantic games.
- Did you know about Pascal's Triangle before? Did you know about linear and binary search before?
- What kind of change should be made to the system to enhance your learning experience.

Questions?

Thank You!

- minVertexBasisSize() := ∀g ∈ Graphs : ∃n ∈ N : vertexBasisSize(g, n) ∧ ¬ vertexBasisSize(g, n−1)
- vertexBasisSize(g ∈ Graphs, n ∈ N) := ∃b ⊆ nodes(g) : basis(g, n, b)

Teams

Playing By Distance

Estimating User Strength: Simple Approach

$$
\begin{aligned}
Wins_{SM}(U_i) &= \sum_j Payoff(U_i, U_j) \\
Losses_{SM}(U_i) &= \sum_j Payoff(U_j, U_i) \\
Str_{SM}(U_i) &= Wins_{SM}(U_i)/(Wins_{SM}(U_i) + Losses_{SM}(U_i))
\end{aligned}
$$

```
W\!ins_{SM}(U_i) &=& \sum_{j} P\!ayo\!f\!\!f\!\,({U_i},{U_j}) \\
Losses_{SM}(U_i) &=& \sum_{j} P\!ayo\!f\!\!f\!\,({U_j},{U_i}) \\
Str_{SM}(U_i) &=& W\!ins_{SM}(U_i)/(W\!ins_{SM}(U_i)+Losses_{SM}
```

Estimating User Strength: Iterative Approach

$$
\begin{aligned}
Str^{0}_{IT}(U_i) &= Str_{SM}(U_i) \\
Wins^{(k)}_{IT}(U_i) &= \sum_j Payoff(U_i, U_j) * Str^{(k-1)}_{IT}(U_j) \\
Losses^{(k)}_{IT}(U_i) &= \sum_j Payoff(U_j, U_i) * (1 - Str^{(k-1)}_{IT}(U_j)) \\
Total^{(k)}_{IT}(U_i) &= Wins^{(k)}_{IT}(U_i) + Losses^{(k)}_{IT}(U_i) \\
Str^{(k)}_{IT}(U_i) &= \begin{cases} 0.5, \text{if } Total^{(k)}_{IT} = 0 \\ Wins^{(k)}_{IT}(U_i)/Total^{(k)}_{IT}(U_i), \text{o/w} \end{cases}
\end{aligned}
$$

- Winning against a strong user results in a large gain. Losing against a strong user results in a small hit.

```
Str_{IT}^{0}(U_i) &=& Str_{SM}(U_i) \\                    56
W\!ins_{IT}^{(k)}(U_i) &=& \sum_{j} P\!ayo\!f\!\!f\!\,({U_i},{U_j}) * Str_{IT}^{(k-1)}(U_j) \\
```