Crowdsourcing Theorem Proving via Natural Games*

Naveen Sundar Govindarajulu[1] and Selmer Bringsjord[2]

[1] Yahoo
Sunnyvale, CA
[2] RAIR Lab
Dept. of Computer Science
Dept. of Cognitive Science
Rensselaer Polytechnic Institute

1 Introduction

Despite the science of modern formal reasoning being more than a century old, mechanized formal reasoning is nowhere near what expert human reasoners (formal and informal) can achieve. Meanwhile, there is a steadily increasing need for automated theorem proving in various fields of science and engineering. Proof discovery and verification in science and mathematics [2,3,4,5], formal verification for hardware and software [6,7] and logic-based AI [8,9,10] are all based on formal theorem proving.

We present initial results from a project on augmenting automated theorem provers with crowdsourced theorem proving through games. The project was started based on the following two seemingly unrelated observations:

Observation 1 (Observation 1) *Non-trivial formal theorem proving requires **insight** into the structure of the problem being solved. (This insight is provided by the problem domain's experts.)*

What do we mean by insight? While it is hard to quantify this, a rough characterization follows. For instance, say we are modelling a domain (e.g. standard arithmetic over $\mathbb{N}$ or biological processes) with a set of axioms $\mathcal{A}$ and we are interested in a conjecture Γ. Assume $\mathcal{A}$ can derive Γ, that is Γ is a theorem of $\mathcal{A}$. Also, assume that the smallest proof of Γ from $\mathcal{A}$ is of length L via a standard proof calculus ρ, that is $\mathcal{A} \vdash^{\rho}_{L} \Gamma$.

One form of insight might be through possession of a library of lemmas by experts.

Definition 1 (Insight via Lemmas). *An expert in the domain might possess insight in the form of a set of lemmas $\mathcal{L}$ (such that $\forall \phi \in \mathcal{L} : \mathcal{A} \vdash \phi$), that enables the expert to derive Γ from $\mathcal{A} \cup \mathcal{L}$ using a proof of length L' with $L' \ll L$.*

Of course, experts could also possess insight not just through lemmas but also through common patterns of reasoning that can be reused across problems.

* Note: While this submission draws partly from the first author's dissertation [1], it contains results and discussions not present in the dissertation. The new data can be download here: https://s3.amazonaws.com/www.catabotrescue.com/data/AnonoynmizedLevelCompletion.json.

Definition 2 (Insight via an Extended Proof Calculus). *An expert in the domain might possess insight in the form of an extended proof calculus* ρ' *that is* ***sound*** *(that is forall* Φ *and* ψ*, if* $\Phi \vdash^{\rho'} \phi$ *then* $\Phi \models^{\rho'} \phi$*).*

This proof calculus enables the expert to derive Γ *from* $\mathcal{A}$ *through* ρ' *using a proof of length* L' *with* $L' \ll L$*, that is* $\mathcal{A} \vdash^{\rho'}_{L'} \Gamma$.

Insight through lemmas is usually domain and problem dependent, and insight through extened proof calculi usually transfers between domains and problems.

Unless such insight is provided to a theorem prover in some form, such problems can be very hard to solve fully automatically. In practice, help is provided by human experts via lemmas [5], proof tactics [11], proof methods [12] etc. This requires that the human experts understand not only the problem domain, but also the formalization of the domain, and other formal tools that might be needed for the task. But for a lot of domains, e.g. hardware verification, there are not enough experts trained in both theorem proving and the problem domain to help with proving non trivial problems.

Observation 2 (Observation 2) *Lay games*[a] *(such as chess or sudoku) which require the player to think logically or deeply are in fact formal reasoning schemes in disguise.*

[a] A *lay game* is simply any game which does not require extensive formal training on the order of what is required for someone to prove formal theorems in logic.

For example, even if we rename all the pieces in chess (or any similar game), it is not wrong to assume that players with some acclimatization will revert to their skills levels in unmodified chess in a short duration compared with the time they took to learn chess from scratch.

The two observations lead us to the question of whether hard formal problems can be cast into a form, for example a natural game like chess, which would remove the need for domain knowledge by completely abstracting away from the original domain. Ideally, such a game should be able to exploit a human player's insight into the structure of the game. This insight should then be automatically transferrable into solving the problem.

We present initial promising results from the *Uncomputable Games* project (first proposed in [13]) aimed at designing and implementing such natural games. While we have designed games for first-order logic, we present initial promising results for propositional logic. The full set of games for first-order natural deduction and first-order resolution theorem proving can be found in [1]. We focus on these proof calculi for the following reasons: resolution is the proof calculus of choice for industrial strength theorem provers, and natural deduction is the proof calculus of choice for formal proofs by humans [12]. The overall vision for this project is summarized in Figure 1.

2 Related Work

While [1] contains a more detailed discussion of related work, we quickly go through a couple of two important related systems. While *General Game Playing* [14] uses the

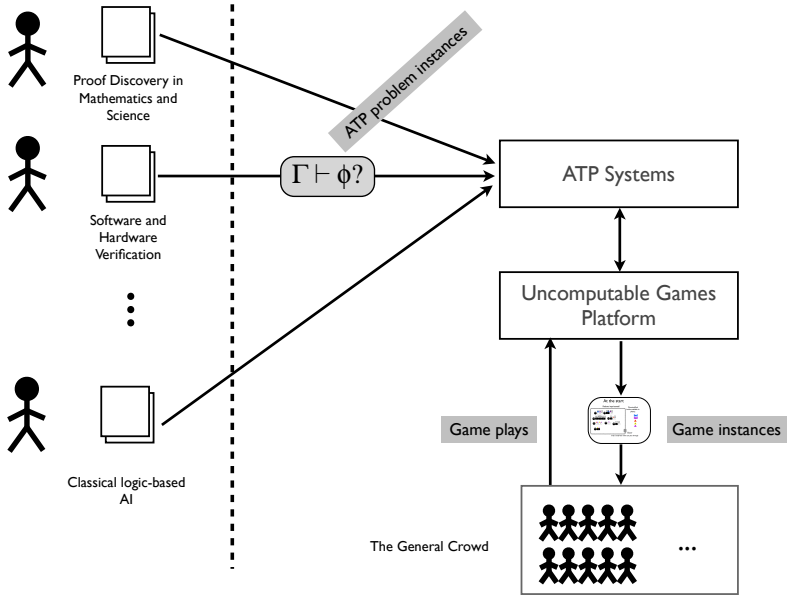


Fig. 1: **The Uncomputable Games Project Architecture**. The abstract architecture for crowdsourced theorem proving. Gamers are not visible to the theorem proving users and gamers do not know about the problems they are solving.

machinery of first-order logic (FOL) to specify the games, the games themselves are neither designed to capture FOL nor to be used in crowd-sourced theorem proving. Since each state in such games has only a finite number of moves, such games are at the propositional level and cannot capture theoremhood in FOL. Tiling problems [15] are undecidable and capture theoremhood in FOL. This is an attractive property for us, as such systems are powerful enough to capture FOL, while also being in the form of a lay game. Unfortunately we cannot use tiling systems for the following reason. Tiling problems generally ask whether an infinite plane can be tiled or not with a given set of tiles. We cannot use such systems as there is no clear notion of certification for an answer. Affirmative answers have as proof either infinitely tiled planes or a proof in some non-tile system (usually informal proofs in mathematical English or formal proofs in FOL), and negative answers don't have a proof using only tiles (even an infinite number of them) and require a proof in some other system.

3 Deriving Game Criteria

Given that we want to have games for crowdsourcing theorem proving, we present some general criteria to aid in the design of games that might help us in this goal. If we are lax with our requirements, we end up with trivial games that are not useful for us. One such game could just present a logic problem "$\Gamma \vdash \phi$?" directly as a game. Another such

game could present to the player a non-deterministic Turing machine that would accept the string $\langle \Gamma, \phi \rangle$ only if $\Gamma \vdash \phi$. At any stage of the computation, a non-deterministic Turing machine has one or more choices.[3] A game could then challenge the player to select choices which result in an accepting state given some input! It is obvious that approaches such as these would not help us.

The overarching requirement that players with no knowledge of formal logic be able to prove theorems by playing games helps us derive some general criteria such games should satisfy. We need a few simple definitions to make our criteria more clear.

3.1 A Simple Formalization

We need a simple vocabulary to help us talk clearly about the aspects of games we are interested in. Some straightforward definitions are given below:

Game Objects Game objects are all the objects that a player can interact with and manipulate in the game. The universe of all game objects is denoted by G. Note that each game object on its own contains enough information to assemble the game at any point in time when combined with all other game objects. The game state at any point in time is a function of the game objects.

Game State The game state at any point of time is just a set of game objects. But not all sets of game objects might correspond to a meaningful configuration of the game. A *game state* is any valid set of game objects.

Game A game $\mathcal{G}$ is a quadruple $\langle G, \gamma \subseteq G \times 2^G, I, F \rangle$ composed of the universe of all game objects G, an operation γ specifying valid operations with the game objects, the initial game states $I \subseteq 2^G$ and the final states $F \subseteq 2^G$.

Game Instance The initial game states are also called *game instances*.

Game Size The game size $|g|$ of a game instance g is simply the number of game objects in that game instance.

Game Play A terminating game play for a game instance g_1 is sequence of game states $g_1, g_2, \ldots, g_n$ such that $g_{i+1} = g_i - \{u_{g_i}\} \cup \nu(u_{g_i})$ where u_{g_i} is some game object in g_i and $\nu(u_{g_i})$ is a set of game objects such that $\gamma(u_{g_i}, \nu(u_{g_i}))$ holds. Game play proceeds by replacing game objects with one or more other game objects.

Captures FOL A game captures **FOL** or theorem-proving in a proof calculus ρ in **FOL** iff every problem $\Gamma \vdash \gamma$ is isomorphic with exactly one game instance $g_{\Gamma \vdash \gamma}$ and every proof for the problem is isomorphic with every terminating game play for the instance. $g_{\Gamma \vdash \gamma}$ is called the *game instance for* $\Gamma \vdash \gamma$.

3.2 Game Criteria for Naturalness and Usefulness

Any game designed that seeks to help us should satisfy the following criteria:

1. **Independently Incentivized**: The games themselves should be captivating and engaging enough that users play the games of their own accord without the incentive of good theorems being proved. This sets us apart from games such as FoldIt where players

[3] A non-deterministic Turing machine m is said to accept an input string s if there is a sequence of choices that results in an *accepting state* s_a.

deal with the problem domain directly. An incentive structure is usually superimposed on the domain.

2. **No Knowledge Needed**: In order to play the games, the players need not have any knowledge of first-order logic and the capturing of first-order theorem proving in the game. To play, users need to know only the rules of the game.
3. **Non Linguistic**: Games that are linguistic in nature generally have a smaller user base. This is a vague criterion which will be sharpened through the course of this project. The general idea is that a player should not have to understand, for example, how Turing machines work in order to play the game. One way to sharpen this criterion is to stipulate that γ has physical correlates or interpretations in naïve physics (e.g. dropping objects, spatially locking objects etc.).
4. **Readable Proof**: If a game play instance shows us that a theorem ϕ can be proved from premises Γ, the game play instance should also provide us with a readable proof ρ. This rules out tiling problems [15] as there is not clear notion of a proof.
5. **Tractable Representation**: Given a problem $\Gamma \vdash \phi$? with logical signature Σ being used in the problem, we can set a bound on the game size. If g is the game instance which represents the problem, then we need to have: $|g| \leq |\Gamma| + 1 + |\Sigma|$. Another condition can be derived if we consider the length of the smallest proof. If the minimum proof length defined as the number of inference rule applications is denoted by $|\Gamma \vdash \phi|$, we require that the instance g for the problem have a terminating game play of length $\|g\|$ such that $\|g\| \leq k * |\Gamma \vdash \phi|$ where k is a non-negative constant.

4 Catabot Rescue Games

The **Catabot Rescue Games** were designed with the above five criteria in mind. The games capture first-order logic with binary resolution and natural deduction. The games are set in a world populated by entities called catabots, which resemble robotic caterpillars or bugs that consume objects in the game for fuel and can reproduce with other catabots. Most of the operations in the game have natural physical correlates, such as dropping objects, dissolving objects, breaking up catabots, etc. No elements of the underlying logic problem are directly shown in the game. A record of the game play can be directly translated into a proof in either binary resolution or natural deduction. The basic design metaphors used in these games could also be utilized to invent and design other games which satisfy the criteria. We present only the propositional subset of one of the games. The full set of games can be found in [1]. Before we present the game, we go through a brief introduction to the **resolution** rule used in theorem proving.

5 Resolution-based Theorem Proving

There are several sound and complete proof calculi for propositional logic with different advantages and strengths. Resolution invented by Robinson [16] is widely implemented in automated provers and logic-programming systems due to its simplicity. While compared with proof calculi adhering close to natural deduction, resolution proofs suffer from being a bit harder for humans to read. Although resolution proofs are hard to read, they have a much simpler formal structure, which makes resolution more amenable to automation. Before we present the game for resolution, we briefly review a formal specification of resolution.

Given two conjunctive normal form clauses $p_1 \vee \ldots \vee p_i \vee \ldots \vee p_n$ and $q_1 \vee \ldots \vee q_j \vee \ldots \vee q_m$, such that $p_i \equiv r$ and $q_j \equiv \neg r$, where r is a propositional variable, the *binary resolution* rule produces:

$$p_1 \vee \ldots \vee p_{i-1} \vee p_{i+1} \vee \ldots \vee p_n \vee q_1 \vee \ldots \vee q_{j-1} \vee q_{j+1} \vee \ldots \vee q_m$$

A proof of $\Gamma \vdash \phi$ is obtained via resolution when there is a series of binary resolution steps resulting in the empty clause denoting **false** using only the clausal form (disjunctive normal form) of sentences in $\Gamma \cup \{\neg\phi\}$ and any sentence that was generated by applications of the rule. We now present the Catabot Rescue Game designed to capture propositional resolution.

6 Catabot Rescue Game 0

The*Catabot Rescue Game 0*, CRG^0, captures propositional resolution. The catabots in the game are isomorphic with clauses in a propositional language and the game process is isomorphic with the steps in a resolution proof. The game satisfies the criteria defined above and adheres to the formalization presented. We start with a description of the game world. The backstory for the game is given below (A similar story is presented to the user along with a figure similar to Figure 4.):

> There are a bunch of catabots imprisoned in a room by a horrible blob. The room has a small door to the outside. None of the catabots are small enough to use the door. Only a body-less catabot with just the head can go through the door! Catabots can reproduce and download their minds to any child's body (or any other catabot). Reproduction has certain rules. Help them escape by mating them in a fashion so as to produce a body-less catabot.

Each catabot has a head and a body composed of zero or more **blocks**. The blocks have a certain number of **limbs** or legs of either positive or negative polarity such as those shown in Figure 2.

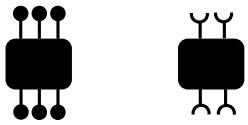
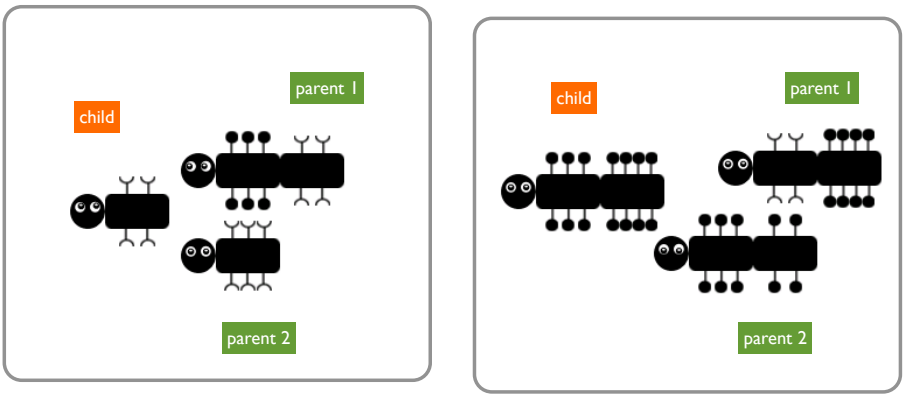


Fig. 2: CRG^0 **Blocks**. Blocks with different polarities.

The catabots represent clauses and the blocks on the body represent different atoms. Let us assume the supply of propositional variables $\{P_1, \ldots, P_k\}$. Then, e.g., P_n would have n limbs with a positive polarity and $\neg P_n$ would have n limbs with a negative polarity. Figure 2 shows literals P_3 and $\neg P_2$.

With this process it is easier to visualize the process of binary resolution operating on two clauses. Two simple examples are shown in Figure 3.



$$\frac{P_3 \vee P_2, \neg P_3}{P_2} \qquad \frac{\neg P_2 \vee P_4, P_3 \vee P_2}{P_3 \vee P_4}$$

Fig. 3: CRG^0 **Mating Example**. Two mating examples. The children are shown on the left side of the parents. The children have all the blocks from the parents, save for the pair of blocks which were *mated* in the parents.

6.1 Mating Rules

Catabot mating rules are very simple and correspond to propositional binary resolution. The rules for catabot reproduction are given below:

Mated Pairs Cancel The user can choose any two catabots for mating. Two catabots can mate if and only if they have the same number of limbs with differing polarities. Such a pair is called a **mated pair**.

Child Catabot The child is a catabot with all the blocks the same, except for the mated blocks in the parents.

Same Blocks Merge If two blocks on different parents are visually the same, they merge into one in the child.

7 Initial Results

An initial industrial-strength implementation of a subset of the operations in CRG^0 has been implemented in a game for iPads and the Web.[4] Figure 4 shows a screenshot of the game.

This initial version has 50 levels out of which 44 were generated automatically (in addition to first six levels which are considered tutorial levels). The generation process

[4] The web version is available at http://compete.catabotrescue.com and the iPad version is available at https://itunes.apple.com/us/app/catabot-rescue-lite/id645249674?ls=1&mt=8. Please note that the iPad version is not compatible with more recent versions of the iPad. We suggest that reviewers use the web version.

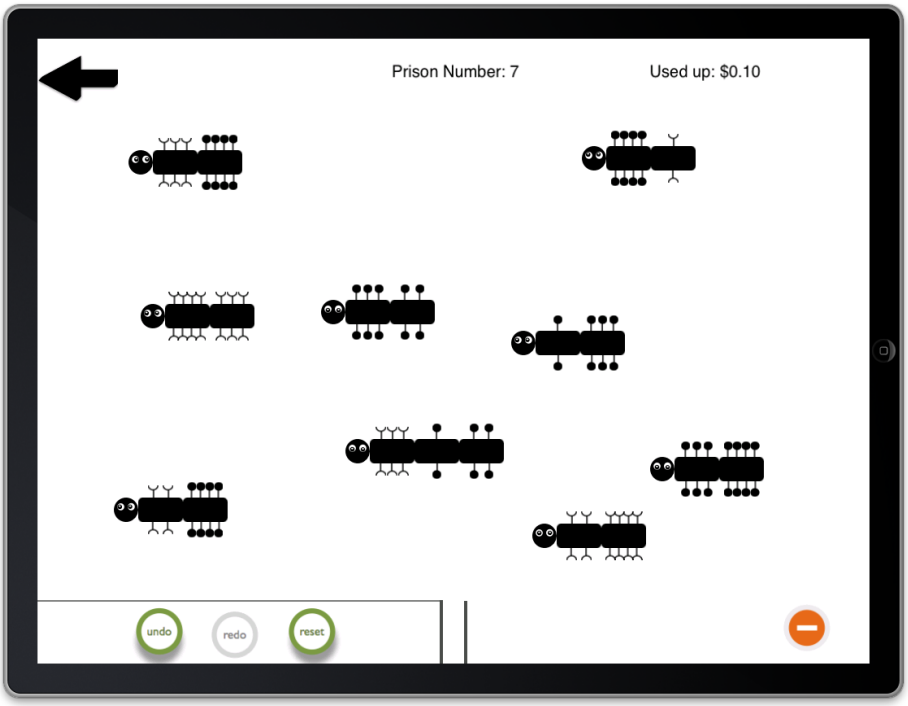


Fig. 4: **Catabot Rescue Game .** A screenshot of Catabot Rescue Game 0 Beginner

of the problems is as follows. We randomly generate a certain number of clauses each having a minimum number *min* and a maximum number *max* of literals. Given that we want n clauses, we sample n times the discrete uniform distribution $[min, max]$ to generate the size of each clause. Then for each clause we uniformly select from the set of all possible literals. We consider only the literals:

$$\{p_1, p_2, p_3, p_4, \neg p_1, \neg p_2, \neg p_3, \neg p_4, \}$$

Most of the clauses generated in this fashion do not result in a contradiction. The Prover9 automated theorem prover [17] was used to filter out sets of clauses which did not lead to a refutation. Prover9's ability to search for multiple proofs was used. Then problems for which the smallest proof was below a certain length were filtered out. Prover9 also reports the depth of the resolution proof tree. Problems with proofs less than four levels deep were removed. This process produces quite complex problems. Though the problems are quite complex and hard, users without any formal training in logic found the levels engaging and were able to solve them. The scoring mechanism for the overall game is in terms of dollar rewards for the user. Upon completion of a level n with k operations, the total reward for the user becomes $\$(n + 100 * 1/k)$.

8 Data from Game Plays

We conducted two experiments with the game. In the first experiment, we released the game to the public and got game play results from mostly players untrained in logic. While we don't have exact numbers on the fraction of the players that were trained in logic, this experiment is supposed to mirror conditions that we might face eventually if the games are to be used on a large scale. In the second experiment, we held a contest in an introductory logic class. The students participating in the class were trained in propositional logic but had no knowledge of resolution.

8.1 Experiment 1: Untrained Humans

Over the course of a few days (less than ten days) since the release of the iPad version of the game, the game was download around 60 times. This resulted in 400 terminating game plays collected from around 15 players. The total number of players is not accurate as users generally hand over played levels to other players and sometimes reset levels to try again and get a better score. We are not interested in metrics which dictate that we have an accurate count of the number of users. Ultimately, when we want to solve very a hard theorem through games, we only need one single player/team to solve the corresponding game instance. Among all the players, only four players were able to solve all the levels. Among these four, we were able to verify that at least two did not have any training in formal logic. Of these two players, one was able to produce smaller proofs than Prover9 for all the 44 levels, save for one. The other players who completed all the levels also produced proofs smaller than Prover9 for some of the levels.

8.2 Experiment 2: Partially Trained Humans

Over the course of a week we had 1766 terminating game plays from around 80 users. Of all these users, 22 completed all the levels.

8.3 Results

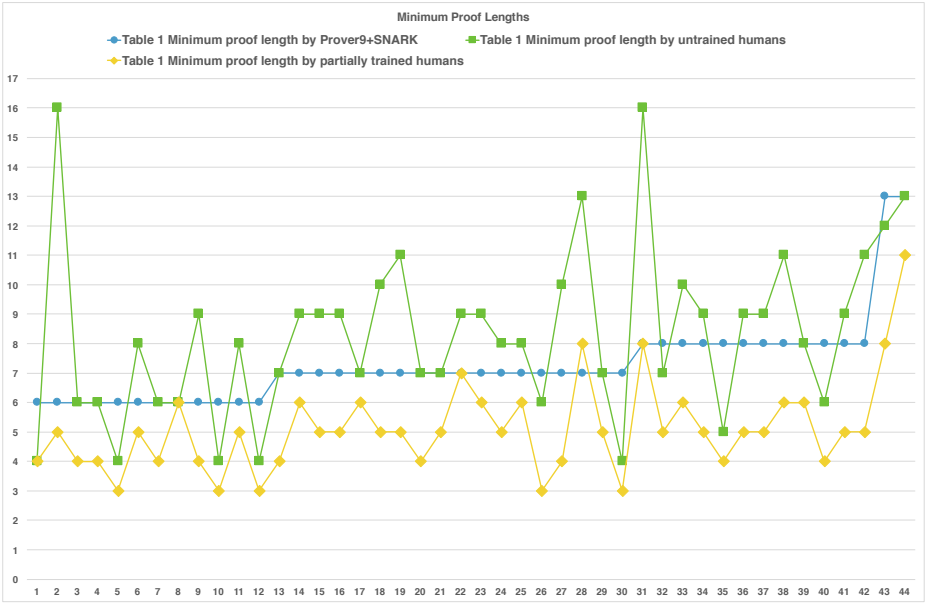


Fig. 5: **Prover9 versus a humans**. Shortest proofs for the 44 levels by humans and Prover9

Figure 5 shows the minimum proof lengths produced by untrained humans, partially trained humans and Prover9 and SNARK combined. Quite surprisingly, minimally trained humans were able to get proofs smaller than the theorem provers.

It should be noted that the two players from the first experiment who did not have any training in formal logic were able to solve problems as hard as the one given below. This problem is represented in Level 50 of the game. We need to derive the empty clause from this given set of clauses:

$$\left\{ \begin{array}{l} P_1 \vee P_3 \vee \neg P_2 \vee P_4, P_4 \vee \neg P_2 \vee \neg P_1, \neg P_4 \vee \neg P_3 \vee \neg P_2 \vee P_1, \neg P_2 \vee P_4 \vee \neg P_3, \\ P_2 \vee \neg P_1, P_3 \vee P_2, \neg P_4 \vee P_1, P_1 \vee P_2, P_1 \vee \neg P_4 \vee P_3 \vee \neg P_2 \end{array} \right\}$$

Illustration of a Human-Dominated Level We now walk through a simple level in which it is easy to see how humans performed better than the ATP. The level we illustrate is Level 7 in the game. The logic problem represented is given below:

$$\left\{ \begin{array}{l} \neg P_1 \vee \neg P_3, P_4 \vee P_3, P_2 \vee \neg P_4, \neg P_3 \vee \neg P_2, \\ P_2 \vee P_4, P_3 \vee \neg P_4, P_3 \vee \neg P_1 \vee \neg P_2, \neg P_3 \vee P_1 \end{array} \right\}$$

Figures 6, 7, 8, and 9 show one possible proof of only four steps. Prover9 was unable to find any proofs with less than six binary resolution steps. Though Prover9's configuration for the maximum number of proofs to search for was set at 50, Prover9 generated only two proofs. The two proofs are shown in Figure 10.

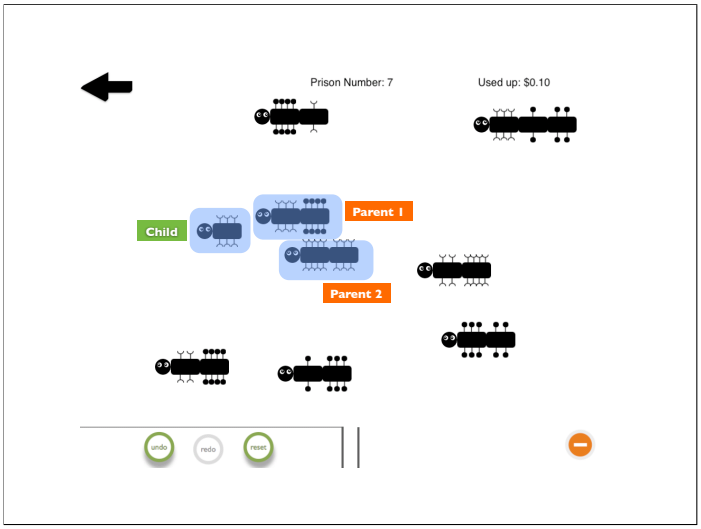


Fig. 6: **Step 1 of the Shortest Human Proof for Level 7**. Catabots corresponding to the premises $P_3 \vee \neg P_4$ and $P_4 \vee P_3$ combine to give the catabot for P_3.

We also ran the SNARK [18] ATP on this problem. While SNARK does have the provision to search for multiple proofs or produce only binary resolution proofs, it is easy to see that proof by SNARK shown in Figure 11 does not translate into a resolution proof smaller than 5 steps.

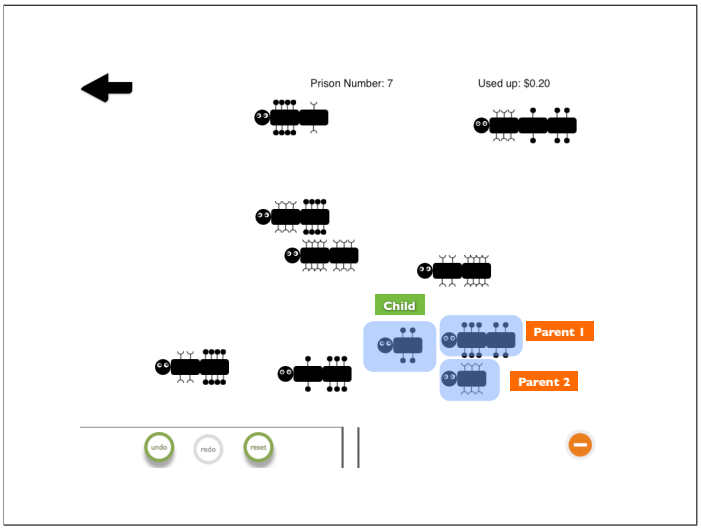


Fig. 7: **Step 2 of the Shortest Human Proof for Level 7**. Use the catabot corresponding to P_3 from the previous step with the catabot for premise $\neg P_3 \vee \neg P_2$ to get $\neg P_2$.

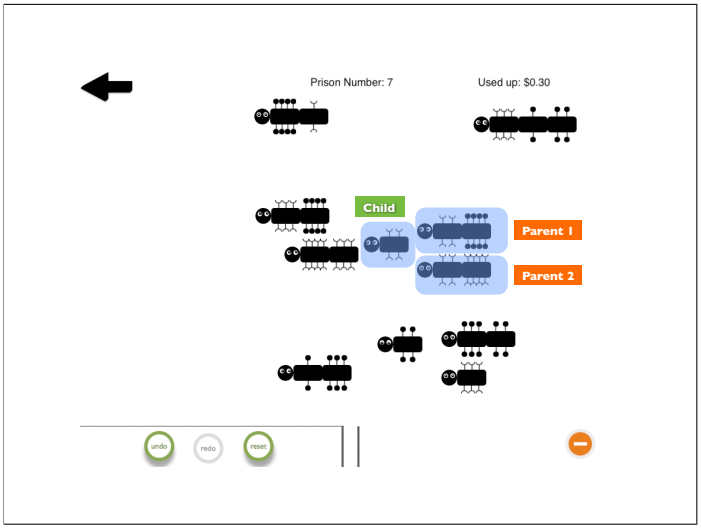


Fig. 8: **Step 3 of the Shortest Human Proof for Level 7**. Combine catabots for premises $P_2 \vee \neg P_4$ and $P_2 \vee P_4$ to get the catabot for P_2. Note: the two blocks for P_2 are automatically merged into one block in the child catabot.

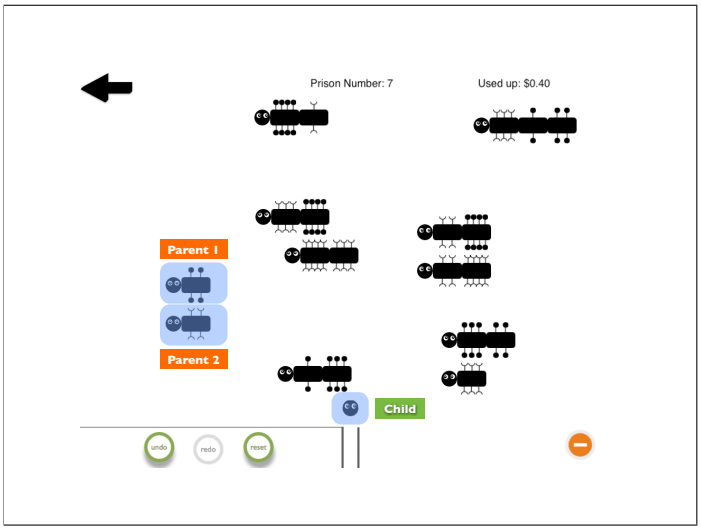


Fig. 9: **Step 4 of the Shortest Human Proof for Level 7** . Finally, mate catabots for P_2 and $\neg P_2$ obtained in the last two steps. This gives us a body-less catabot representing **false**. This catabot then escapes!

```
% Length of proof is 11.
% Level of proof is 4.
% Maximum clause weight is 2.000.
% Given clauses 9.

2 -P1 | -P3.  [assumption].
3 P4 | P3.  [assumption].
5 -P3 | -P2.  [assumption].
6 P2 | P4.  [assumption].
7 P3 | -P4.  [assumption].
9 -P4 | P1.  [assumption].
10 -P2 | P4.  [resolve(5,a,3,b)].
11A P4 | P4.  [resolve(10,a,6,a)].
11 P4.  [copy(11A),merge(b)].
12 P1.  [resolve(11,a,9,a)].
13 P3.  [resolve(11,a,7,b)].
15A -P3.  [resolve(12,a,2,a)].
15 $F.  [resolve(13,a,15A,a)].
```

(a) Proof 1

```
% Length of proof is 10.
% Level of proof is 4.
% Maximum clause weight is 2.000.
% Given clauses 9.

3 P4 | P3.  [assumption].
4 P2 | -P4.  [assumption].
5 -P3 | -P2.  [assumption].
6 P2 | P4.  [assumption].
7 P3 | -P4.  [assumption].
10 -P2 | P4.  [resolve(5,a,3,b)].
11A P4 | P4.  [resolve(10,a,6,a)].
11 P4.  [copy(11A),merge(b)].
13 P3.  [resolve(11,a,7,b)].
14 P2.  [resolve(11,a,4,b)].
16A -P2.  [resolve(13,a,5,a)].
16 $F.  [resolve(14,a,16A,a)].
```

(b) Proof 2

Fig. 10: **Prover9's two proofs for Level 7**: Prover9's two proofs for Level 7 are longer than the shortest proof found by humans.

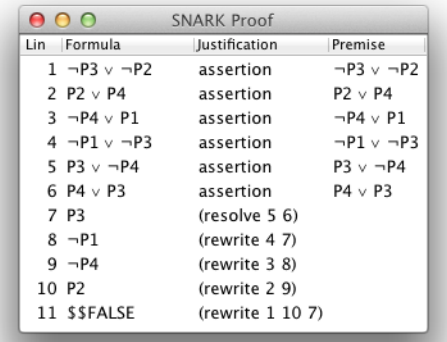


Fig. 11: **SNARK on Level 7**. The proof from the SNARK ATP is still longer than the shortest proof found by humans.

9 Observations

While nowhere close to a full validation of our goal, these initial results provide us with some promise that such games could enable crowds of untrained humans to help machines when it comes to solving hard problems. It is quite notable that *humans without much formal training in logic can produce proofs shorter than those produced by machine in a calculus specialized for machine reasoning.* This is remarkable, because, in addition to propositional theorem proving being hard, many associated problems in propositional logic are hard. For example, determining whether there is a proof of a certain length for a tautology, determining the length of the shortest proof for a given tautology, and finding the shortest proof given the length of the shortest proof are all believed to be quite hard [19,20]. These results, while humble, indicate that theorem provers might be able to benefit from untrained humans. Future experiments would involve a more extensive set of problems (e.g. bins of problems of the same minimum proof length) to get more statistically useful conclusions comparing human performance with that of machines.

10 Data and Source Code

1. The data from Experiment 2 can be download from here: https://s3.amazonaws.com/www.catabotrescue.com/data/AnonoynmizedLevelCompletion.json.
2. The cross platform source code for the game is available here: https://github.com/naveensundarg/catabotrescue-unity
3. The game can be played here: http://compete.catabotrescue.com

References

1. Govindarajulu, N.S.: Uncomputable Games: Games for Crowdsourcing Formal Reasoning. PhD thesis, Rensselaer Polytechnic Institute (RPI) (2013) Available at: https://s3.amazonaws.com/naveensundarg/Dissertation.pdf.
2. Naumowicz, A., Kornilowicz, A.: A Brief Overview of Mizar. In Berghofer, S., Nipkow, T., Urban, C., Wenzel, M., eds.: Theorem Proving in Higher Order Logics. Volume 5674 of Lecture Notes in Computer Science (LNCS). Springer, Berlin (2009) 67–72
3. Wos, L.: The Legacy of a Great Researcher. In Bonacina, M.P., Stickel, M.E., eds.: Automated Reasoning and Mathematics: Essays in Memory of William McCune. Springer, Berlin (2013) 1–14
4. Govindarajulu, N.S., Licato, J., Bringsjord, S.: Small Steps toward Hypercomputation via Infinitary Machine Proof Verification and Proof Generation. In Mauri, G., Dennunzio, A., Manzoni, L., Porreca, A., eds.: Unconventional Computation and Natural Computation. Volume 7956 of Lecture Notes in Computer Science (LNCS). Springer, Berlin (2013) 102–112
5. Govindarajulu, N.S., Bringsjord, S., Taylor, J.: Proof Verification and Proof Discovery for Relativity. Synthese **192** (2015) 2077–2094
6. Dean, D.: Crowd Sourced Formal Verification (CSFV) (2013) Retrieved on July 26, 2013. http://www.darpa.mil/Our_Work/I2O/Programs/Crowd_Sourced_Formal_Verification_(CSFV).aspx.
7. Almeida, J.B., Frade, M.J., Pinto, J.S., de Sousa, S.M.: Rigorous Software Development: An Introduction to Program Verification (Undergraduate Topics in Computer Science). Springer, Berlin (2011)
8. Charniak, E., McDermott, D.: Introduction to Artificial Intelligence. Addison-Wesley, Reading (1985)
9. Genesereth, M., Nilsson, N.J.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann, Los Altos (1987)
10. Bringsjord, S.: The Logicist Manifesto: At Long Last Let Logic-Based AI Become a Field Unto Itself. Journal of Applied Logic **6**(4) (2008) 502–525
11. Delahaye, D.: A tactic language for the system coq. In: Logic for Programming and Automated Reasoning, Springer (2000) 85–95
12. Arkoudas, K.: Denotational Proof Languages. PhD thesis, MIT (2000)
13. Govindarajulu, N.S.: Uncomputable Games: Toward Crowd-sourced Solving of Truly Difficult Problems. In: Turing Centenary Conference 2012:How the World Computes (Abstracts of Informal Presentations). CiE (2012) Page 63 http://www.mathcomp.leeds.ac.uk/turing2012/WScie12/Images/abstracts-booklet.pdf.
14. Genesereth, M., Love, N., Pell, B.: General Game Playing: Overview of the AAAI Competition. AI Magazine **26**(2) (2005) 62–72
15. Berger, R.: The Undecidability of the Domino Problem. Memoirs of the American Mathematical Society **66** (1966) 1–73
16. Robinson, J.A.: A Machine-Oriented Logic based on the Resolution Principle. Journal of the ACM **12**(1) (1965) 23–41
17. McCune, W.: Prover9 and Mace4 (2013) Retrieved on April 25, 2016. http://www.cs.unm.edu/~mccune/prover9/.
18. Stickel, M.E.: SNARK - SRI's New Automated Reasoning Kit (2008) Retrieved on April 25, 2016. http://www.ai.sri.com/~stickel/snark.html.
19. Alekhnovich, M., Buss, S., Moran, S., Pitassi, T.: Minimum Propositional Proof Length is NP-Hard to Linearly Approximate. Journal of Symbolic Logic **66**(1) (2001) 171–191
20. Buss, S.R.: Some Remarks on Lengths of Propositional Proofs. Archive for Mathematical Logic **34**(6) (1995) 377–394