CrowdMine: Towards Crowdsourced Human-Assisted Verification

Wenchao Li
Sanjit A. Seshia
Somesh Jha

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-121
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-121.html

May 25, 2012

Acknowledgement

This research was supported in part by NSF grant CNS-0644436, an Alfred P. Sloan Research Fellowship, and the Gigascale Systems Research Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.

CrowdMine: Towards Crowdsourced Human-Assisted Verification

Wenchao Li
UC Berkeley
wenchaol@eecs.berkeley.edu

Sanjit A. Seshia
UC Berkeley
sseshia@eecs.berkeley.edu

Somesh Jha
UW Madison
jha@cs.wisc.edu

ABSTRACT

We propose the use of crowdsourcing and human computation to help solve difficult problems in verification and debugging that can benefit from human insight. As a specific scenario, we explain how non-expert humans can assist in the verification process by finding patterns in portions of simulation or execution traces which are represented as images. Such patterns can be used in a variety of ways, including assertion-based verification, improving coverage, bug localization, and error explanation. Several related issues are discussed, including privacy and incentive mechanisms.

Categories and Subject Descriptors

B.7.2 [**Design Aids**]: Verification; H.1.2 [**User/Machine Systems**]: Human factors

General Terms

Algorithms, Verification, Human Factors

Keywords

Specification, verification, crowdsourcing, human computation

1. INTRODUCTION

The field of electronic design automation (EDA), in general, and formal verification, in particular, has relentlessly pushed for automation. For several problems, this is indeed the right strategy. But for many problems human insight and involvement remain invaluable. Consider, for example, the process of verifying a design. First of all, one needs to write a specification, typically in the form of properties (assertions) or a reference model. Second, one must create an environment model, typically in the form of constraints on the inputs or a state machine description. Next, one runs the verifier, such as a model checker, which is usually thought of as a "push-button" technique. While this is largely true, human insight is not entirely absent; e.g., one might need to supply hints to the verifier in the form of suitable abstraction techniques or (templates for) inductive invariants. If the verifier returns with a counterexample trace, one must debug the design by localizing the cause of error in time (relevant part of the trace) and space (relevant part of the design). Finally, the process of repairing the design to eliminate the bug is also one that needs human input. To summarize, even after decades of work on automating the verification process, we continue to need human insight in a variety of tasks, including writing specifications, creating models, guiding the verification engine, debugging and error localization, and repair.

This paper takes the position that while we cannot completely remove human insight from the verification process, we can change the way humans provide insight to the verifier. Today, such input typically comes from expert verification engineers, trained in the tools of their field. But such experts are few and expensive. And even experts have a hard time answering questions such as: When are we done verifying? Have we written enough properties? Where is the bug? And so on. We contend that the experts and automated tools can be assisted in the verification process by a large crowd of non-expert humans performing simple and repetitive tasks. Each task involves a pattern recognition or other cognitive operations that humans are typically good at. The main technical challenges are to identify steps in the verification process where human insight is critical, find ways to transform these steps into tasks that non-expert humans can perform, and combine the results to resolve those steps in the verification process. As preliminary evidence to show that these challenges can be met, we present a system called CrowdMine for finding specifications from traces based on pattern recognition by humans.

The idea of tapping into a crowd of humans to assist in a computational task is not new. *Crowdsourcing* is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call [7]. *Human computation* is a paradigm for utilizing human processing power to solve problems that computers cannot yet solve [15]. (See Quinn and Bederson [12] for a more detailed description of these and related terms.) Our proposal is to use a combination of crowdsourcing and human computation to improve the state-of-the-art in verification. The availability of tools like Amazon's Mechanical Turk [1] and TurKit [10] make such a combination easier to deploy today.

In recent years, others have also advocated the use of crowdsourcing and human computation in design and verification, both for hardware and software. DeOrio and Bertacco [5] propose having humans assist in solving NP-complete problems arising in EDA, such as Boolean satisfiability (SAT) solving. Schiller and Ernst [13] propose the use of crowdsourcing and human computation for solving problems in software engineering, including software verification. The important difference between our proposal and these works is that we target steps in the verification process that *already* require human input, and which we think are unlikely to be automated entirely (similar to hard AI problems in the class of passing the Turing test, but unlike many NP-hard problems). We seek to leverage crowdsourcing and human computation to scale up the productivity in these steps manyfold.

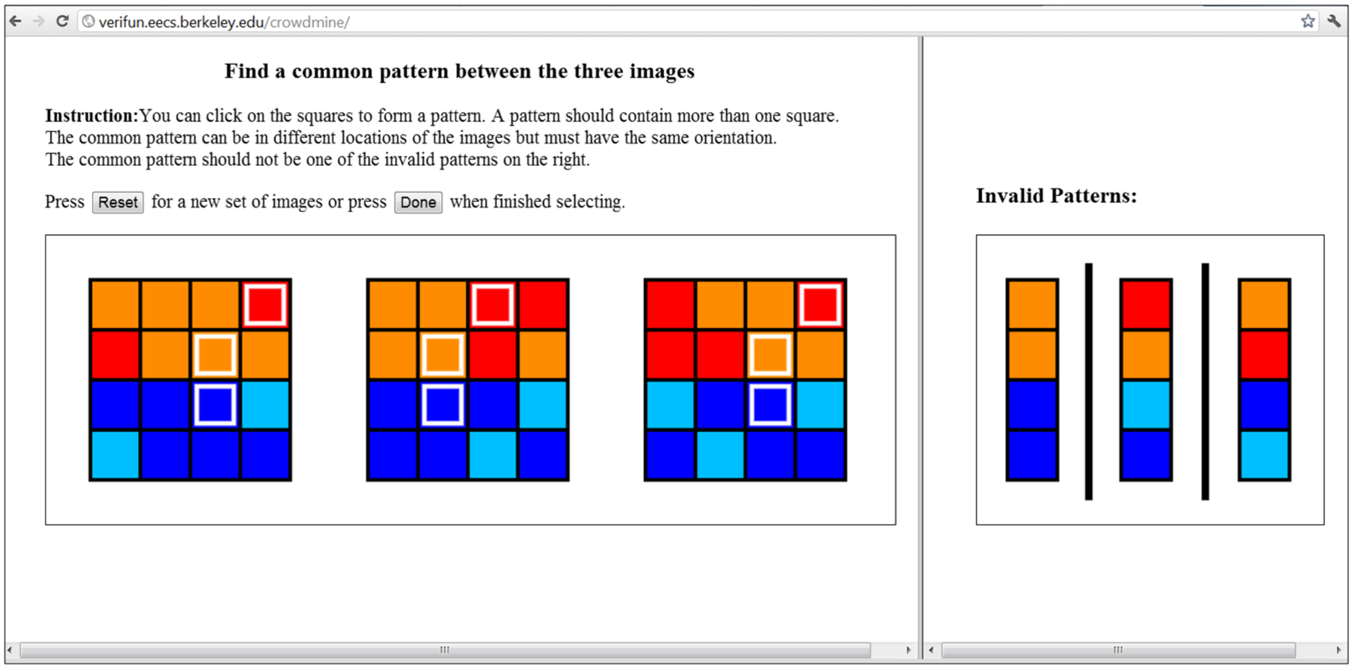
To summarize, this paper makes the following contributions:

- Advocate the use of crowdsourcing and human computation for sub-tasks in verification that require human insight;
- Demonstrate the idea through CrowdMine, a novel game devised for finding patterns from system traces that can suggest likely specifications (Section 2), and
- Sketch out the landscape of similar applications (Section 3).

This technical report is an extended version of the conference publication at DAC'12 [9].

2. CROWDMINE

Many existing behavioral or specification mining techniques rely on the use of templates [6, 8]. Hence, it is the user's responsibiltiy to come up with a good set of templates. This process requires expert insight and is often incomplete. In fact, often times a verification engineer would face the problem of a coverage gap or not knowing if he has verified enough properties of the design. Crowd-


verifun.eecs.berkeley.edu/crowdmine/

Find a common pattern between the three images

Instruction: You can click on the squares to form a pattern. A pattern should contain more than one square.
The common pattern can be in different locations of the images but must have the same orientation.
The common pattern should not be one of the invalid patterns on the right.

Press Reset for a new set of images or press Done when finished selecting.

Invalid Patterns:

Figure 1: Example game interface: the task is to find a pattern that is common in the three images but is not one of the invalid patterns.

Mine is a specification mining game designed to address these issues. The main idea of CrowdMine is that a trace can be visualized as a video or a sequence of images, such that each segment (sub-trace) of the trace is a 2D image. We observe that humans have an innate ability to recognize patterns of similar forms in images. For example, even a young kid can quickly identify the wheels across images of different kinds of vehicles. Crowdmine first transforms segments of a trace into 2D images and then display a small subset of them to a non-expert crowd in the form of a puzzle game. Under the hood, it uses the identified common patterns to infer likely specifications. We have designed and deployed such as a game, shown in Figure 1, that incorporates these ideas.

2.1 Game Design

Game Design:

1. The player is presented with a set of three images along with three other "invalid" patterns. These patterns represent specifications that are either already known or have been identified by automatic techniques. Each pattern is a collection of squares which are not necessarily adjacent.
2. The player is asked to identify a pattern that is common in the three images but does not match any of the invalid patterns given. The patterns can be located in different positions of the three images but must have the same orientation. Additional constraints for the pattern may also be specified.
3. The player can select a candidate pattern by clicking on the individual squares of any of the image. After he has finished selecting, he can click on the button "Done" to register the pattern. If the identified pattern is indeed a common pattern across the three images, it will be stored in a database. It the pattern is already present in the database, its count will be incremented.
4. An optional time limit of T (e.g. 30) seconds is enforced for each play. Also, points may be awarded for the player.

Backend:

1. First, we sample a small set of n (e.g. 4) signals of interest and assign them an arbitrary order. Next, we randomly select three segments of k (e.g. 4) cycles in the trace and project them onto the n signals. This generates the three sub-traces for the game, where rows represent signals and columns represent cycles in the sub-traces.
2. A template-based pattern mining algorithm is first executed to find common patterns that exist in these two sub-traces. Three of these mined patterns are selected in random as invalid patterns. Alternatively, known patterns (such as previous specifications produced by CrowdMine which are also confirmed by the provider of the circuit) are displayed as invalid patterns.
3. All the identified patterns in the database are ranked in decreasing order of their counts. The top ranked patterns are output as likely specifications.

GUI design:

- Different color ranges are used to encode different types of signals, e.g. input signals vs. output signals.
- Different shades of color are used to encode different signal values, e.g. dark for 0 and light for 1 for binary signals.
- The player can click on a square to select it. The selected square will be highlighted in white borders, as shown in Figure 1.
- When the player clicks the "Done" button, the collection of selected squares in each image is considered as the identified pattern.
- The player can choose to play with a different set of images at any time by clicking the "Reset" button.

We have deployed CrowdMine on `http://verifun.eecs.berkeley.edu/crowdmine/`. The underlying circuit in Figure 1 is a 2-input and 2-output arbiter that implements a round-robin scheme of arbitration. We simulated the arbiter with random inputs for 200 cycles and extracted all distinct sub-traces over the I/O signals with a span of 4 cycles. These sub-traces were then randomly sampled to produce the 2D images used in the game as shown in Figure 1. We use this simple example to illustrate our workflow. The bottom two rows are the two request signals req_0 and req_1. The top two rows are the two response signals $resp_0$ and $resp_1$. Figure 2 shows the color coding scheme for converting a sub-trace to a 2D image.

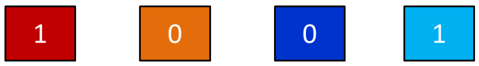


Figure 2: Color coding scheme

We evaluate the effective of CrowdMine based on the 283 plays that we have gathered so far from the Internet players, who are mostly students in the Electrical Engineering and Computer Science department at UC Berkeley. The players are oblivious of the fact that the images represent sub-traces of a circuit. There are a total of 165 distinct patterns identified by the players. The top three ranked patterns have counts 31, 16 and 7 respectively. The "invalid patterns" displayed in Figure 1 are actually the top ranked patterns in the database at the time of the evaluation. The meaning of these patterns, from left to right, are described below.

- (Left): "All signals are low in one cycle, i.e. if there is no request, then both responses signals are low at the same cycle."
- (Center): "When req_1 is high and there is no competing req_0, $resp_1$ is high at the same cycle."
- (Right): "When req_0 is high and there is no competing req_1, $resp_0$ is high at the same cycle."

All three of these patterns correspond to desired behaviors of the arbiter. This means that even though human inputs are very noisy, selection based on a simple ranking metric can still produce useful patterns. In the future, we would like to evaluate how CrowdMine scales to larger circuits and also how to steer the players towards finding more complex patterns.

2.2 Discussion

Privacy. Our design is particularly attractive for companies that value confidentiality because the internals of the circuit are not revealed. For IP protection, the mapping of sub-traces to images should be kept confidential. This mapping include the correspondence of signals in the circuit, the color code, and any additional transformation on the subtraces. Randomization can also be used in selecting sub-traces and the mapping to images. Finally, *secret sharing* methods such as the *threshold schemes* developed by Shamir [14] are particularly relevant in this context.

Incentives. Four mechanisms are possible.

- *Necessity:* Authentication systems such as reCAPTCHA [15] embed queries into a human challenge with partially known answers. Our game design can be augmented for this purpose. For example, two plays are presented to the user in series in which the answer is known for one of the plays.
- *Enjoyment:* Our game design can be viewed as a puzzle game and the player derives enjoyment by solving it. One future direction is to make the game more interactive. For example, the players can challenge one another for identifying the most complex patterns. Deploying the game as a cell phone is another consideration. According to a recent survey, eight in ten adults today in the U.S [11]. are cell phone users, and among them, 43% of them have apps on their phones. In addition, across the two major mobile platforms Android and iOS, games solely constitute 15% and 17% of all active apps [4, 3]. If we assume on average a person spends 10 minutes per day playing cell phone games, that is about 5 to 8 million hours (!) of human intelligence available daily (just in the U.S.). Moreover, among the most popular cell phone games, many of them are casual games that do not involve complicated control or heavy graphics. CrowdMine fits exactly into that category.
- *Brain Exercise:* Lumosity [2] is a web-based company that offers brain training exercises in the form of simple games that aim to enhance the health and function of the brain. One example is a game called "Memory Matrix", where the user is quickly flashed a 2D colored image like the ones in CrowdMine and is asked to recall the positions of squares of a specific color. While the goal of this game and that of CrowdMine are different, the setups are very similar. Finally, Lumosity's community of 20 million registered users are evidential of the possibility of scaling CrowdMine to a large population of players.
- *Profit:* Platforms such as Amazon's Mechanical Turk [1] provide a *for-profit* medium for crowdsourcing any human intelligence task. Additionally, with appropriate monetary reward mechanism, it is possible to deploy CrowdMine as a play-to-get-paid game if the mined specifications turn out to be useful for the EDA companies. We envision this to have disruptive potential on the prevalent pay-to-play culture of the gaming industry today.

Human-Computer Collaboration. It is possible to combine algorithmic techniques with inputs from humans to achieve something better than what can be accomplished by either solely humans or a completely automated approach. In our setting, the human-identified patterns can be further refined to produce the most relevant ones based on feedback from the back-end verification and debugging processes. They can also be used in automated tasks such as bug localization [8].

3. LOOKING AHEAD

We believe several games similar to CrowdMine can be created and applied to a range of applications in verification, debugging, and related areas. For example, one can improve coverage of a design by properties (or tests) by highlighting parts of a trace corresponding to variables not covered by (enough) properties, and users can be provided incentives to find patterns involving those parts. Properties generated by a system like CrowdMine can be hypothesized as auxiliary inductive invariants to speed up verification. Human-observed patterns in spurious counterexamples could potentially enable better abstraction-refinement in model checking. Finally, the process of debugging has similarities to investigating a crime scene (!) — the "crime" is the manifestation of the error (the failure), and one seeks to find a cause-and-effect chain that explains how the failure happened; this analogy suggests a natural game that could be formulated for non-expert humans to assist in debugging.

Acknowledgements. This research was supported in part by NSF grant CNS-0644436, an Alfred P. Sloan Research Fellowship, and the Gigascale Systems Research Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.

4. REFERENCES

[1] Amazon's mechanical turk. `www.mturk.com/mturk/welcome`.

[2] Lumosity. `http://www.lumosity.com/`.

[3] 148Apps.giz. App store metrics. `http://148apps.biz/app-store-metrics/?mpage=catcount`.

[4] AndroLib. Distribution of apps and games in android market. `http://www.androlib.com/appstatstype.aspx`.

[5] A. DeOrio and V. Bertacco. Human computing for EDA. In *Design Automation Conference (DAC)*, pages 621–622, 2009.

[6] M. D. Ernst, J. H. Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao. The daikon system for dynamic detection of likely invariants. *Sci. Comput. Program.*, 69(1-3):35–45, 2007.

[7] J. Howe. Crowdsourcing: A definition. http://crowdsourcing.typepad.com.

[8] W. Li, A. Forin, and S. A. Seshia. Scalable specification mining for verification and diagnosis. In *Proceedings of the Design Automation Conference (DAC)*, pages 755–760, June 2010.

[9] W. Li, S. A. Seshia, and S. Jha. Crowdmine: Towards crowdsourced human-assisted verification. In *Design Automation Conference (DAC)*, page (to appear), June 2012.

[10] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller. TurKit: Human computation algorithms on mechanical turk. In *Proc. 23nd ACM symposium on User interface software and technology (UIST)*, pages 57–66, 2010.

[11] K. Purcell, R. Entner, and N. Henderson. The rise of apps culture, September 2010.

[12] A. J. Quinn and B. B. Bederson. Human computation: A survey and taxonomy of a growing field. In *ACM Conference on Human Factors in Computer Systems (CHI)*, 2011.

[13] T. W. Schiller and M. D. Ernst. Rethinking the economics of software engineering. In *In Workshop on the Future of Software Engineering Research*, pages 325–330, 2010.

[14] A. Shamir. How to share a secret. *Commun. ACM*, 22:612–613, November 1979.

[15] L. von Ahn. *Human Computation*. PhD thesis, Carnegie Mellon University, December 2005.