Emma Tosch · Research Statement

Computer programs automate more tasks than ever before: **data-driven algorithmic decision-making** can inform consequential real-world outcomes in disparate domains such as judicial sentencing, self-driving cars, and massively open online courses. Consequently, an increasing number of **procedures that we traditionally do not think of as computer programs** are now either encoded in software, or interact with software.

Adapting procedures from their manual versions to their digital counterparts can remove some known **errors or threats to validity**, while introducing novel errors or threats that lie exclusively at the intersection of a domain and its expression in software. Novel execution environments such as social media platforms, human computation systems, or even environments for autonomous software agents present new challenges for reasoning about **software correctness**. Fortunately, techniques from the fields of programming languages (PL) and software engineering (SE), such as **static analysis** and **code generation**, can be used outside their typical settings, in the context of these new software domains.

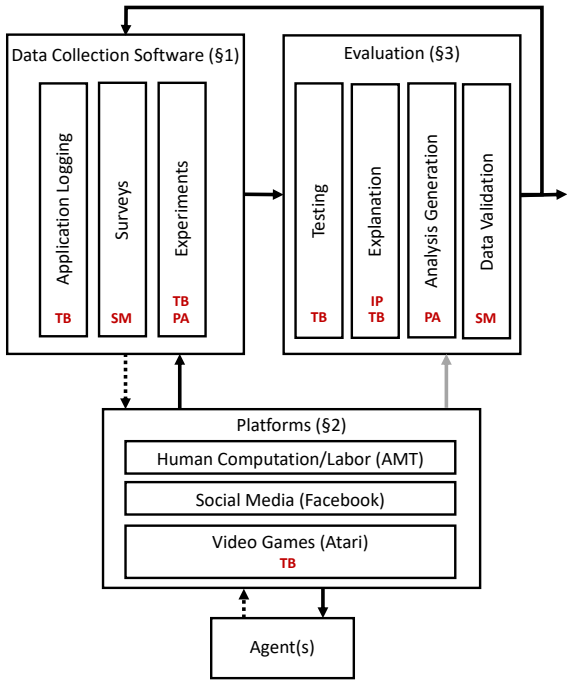

Figure 1: The data analysis pipeline. Solid arrows indicate data flow for all cases; dotted arrows indicate that data only flows in this direction for some applications. One goal of my research is to remove the need to reason about biases induced by the platform in data analysis (hence the grey arrow from "Platforms" to "Evaluation"). **IP**: in progress project, **PA**: PlanAlyzer, **SM**: SurveyMan **TB**: ToyBox.

My research focuses on developing domain-specific tools for stakeholders across a range of fields powered primarily by data: e.g., data science, social science, artificial intelligence, and machine learning. These tools include **domain-specific programming languages (DSLs)**, **runtime systems**, and **testing frameworks**, that enforce domain-specific properties of various tasks. My work has focused on three points in the data analysis pipeline, illustrated in Figure 1: developing techniques for ***automatically ensuring unbiased data collection*** in environments that the researcher does not control (§1), developing ***novel platforms*** informed by principles of data collection and analysis (§2), and designing ***tools that help automate the analysis of data*** (§3). The long-term view of my work is to develop tools that help **democratize data analysis and experimentation**; I discuss connections to **human factors** and **privacy** in my future work (§4).

1 Automatically Detecting and Preventing Biases in Data Collection

All data-driven analysis and decision-making starts with data collection. Data collection tasks can be arranged on a continuum, based on the amount of control a researcher has over the data collection process. Observational studies and machine learning tasks typically use existing data sets whose data generating process and method for selection may not be known (e.g., the "Application Logging" block in Figure 1).

My research focuses on cases where the researcher has some level of control over the data collection process, using platforms such as Amazon's Mechanical Turk (AMT) and Facebook. **There are many potential sources of bias in the data collection process**, but two in particular lend themselves to software-based solutions: *measurement bias*, where the tools we use can have unintended effects on the data we collect, and *selection bias*, where some aspect of our collection apparatus causes the data to not be representative of the intended population.

Measurement bias can be induced by tools that have unintended effects on variables of interest. Online surveys are a common tool for social science researchers to collect data. However, *question wording* and *question order* can bias results. I developed SurveyMan: a **DSL and runtime system for designing, deploying, and debugging web surveys** that can help prevent and diagnose these biases in the data collection process via randomized question selection and ordering, under user-defined constraints (Tosch and Berger, 2014). This work was informed by collaborations with researchers in Linguistics and Labor Studies. We designed the DSL as a **spreadsheet-based language** that would integrate with researchers' existing practices. This work won first place at the **Student Research Competition** at PLDI 2014, and a **Best Paper Award** at OOPSLA 2014.

Selection bias is a major threat to the validity of statistical analyses, and is particularly pernicious in contexts where researchers wish to derive **causal explanations** from data. **Randomized experiments** are the gold-standard for answering causal queries. Web-based experiments can range from *A/B tests* to *contextual bandits experiments* that resemble reinforcement learning (RL). Experiments are now commonly expressed as computer programs via languages such as PlanOut (Bakshy et al., 2014). A DSL can make it easier to write large, complex experiments; however, the price for this expressiveness is that these programs can be tricky to audit and analyze. I developed PlanAlyzer: a **novel static analysis tool** for identifying selection bias in **programmatically-defined experiments** that are written in the PlanOut DSL (Tosch et al., 2019a).

SurveyMan and PlanAlyzer both operate over programs designed to collect data across human participants. However, there are data collection issues present in analyzing large software systems as well, especially when those systems interact with other software such as **autonomous agents**. I am currently collaborating with a colleague on generating sample data for the evaluation of causal inference algorithms using ToyBox (Tosch et al., 2019b) and Starcraft II (Vinyals et al., 2017). This line of research presents unique challenges for data collection due potential mismatches between the data a researcher can record and the data used for analysis, as well as uncertainty over the underlying variability in the data generating process (Clary, Tosch, Foley, and Jensen, 2018).

2 Platform Support for Explaining Behavior

My current work on platform design addresses evaluating and **explaining deep RL** agents. My coauthors and I developed ToyBox, a suite of environments that simulate a subset of games from the Atari benchmark suite (Tosch et al., 2019b; Bellemare et al., 2013). While deep RL agents may seem to have nothing in common with surveys or field experiments, the challenges we face in evaluating these agents are actually quite similar to human computation: both involve **non-inspectable, and potentially non-interpretable, autonomous actors** making long-term decisions in complex environments.

ToyBox environments are **fully parameterized**, supporting **low-overhead intervention** on game state that can be applied mid-game, during training. ToyBox therefore combines the "found" nature of Atari games with the features necessary for intervention. This design allows researchers to treat what were previously individual games, as a family of games – and potential **equivalence class** – that can be varied along several axes. We are starting to use ToyBox to study **transfer learning** in the presence of data drift.

ToyBox has already influenced, and is being used in, two machine learning research projects in my group. Furthermore, an early ToyBox prototype I designed and developed has been used by our collab-

orators at Charles River Analytics (CRA). This prototype supported basic research on the generalization capacity and robustness of a then-state-of-the-art DQN RL agent (Wang et al., 2015; Witty et al., 2018). ToyBox also includes a **behavioral testing framework**, which we discuss in Section 3. This framework would not be possible without platform support for intervention.

Designing RL environments with experimentation in mind is a more constrained version of, for example, re-designing AMT for the dual purposes of supporting a labor market and an experimentation platform. Whether the researcher is interested in human behavior or RL agent behavior, they are fundamentally motivated by questions about **interactions between autonomous actors and an environment**.

3 Automated Experimentation and Evaluation

One major goal of my work is to automate labor-intensive or error-prone tasks within the data analysis pipeline. The pipeline itself is iterative, following the loop illustrated in Figure 1: a researcher fetches data, analyzes that data, and then decides whether to stop or to re-execute the loop. Some algorithms incorporate stopping conditions such as convergence to a point or distribution, or to an equilibrium. However, it is not always clear what the appropriate expression of these stopping conditions are in complex software systems, or how to find them.

Early in my research career, I worked in evolutionary computation. Many of the learning algorithms in this space are **sequential and operate over collections** of objects (e.g., a population of programs). My work proposed and evaluated an approximation of **convergence of distribution** over these data points via order statistics (COSMOS) as a stopping condition for learning, rather than the common practice of a point estimate of the best element in the collection (Tosch and Spector, 2012). Using Spector and Robinson's Push DSL for evolutionary computation (2002) sparked my interest in how software system design and language support can affect outcomes. While the COSMOS work influenced my thinking behind SurveyMan, there is a more direct line between it and the testing framework of ToyBox.

Tests in ToyBox can determine if an agent is learning a **generalized behavior**. Tests are both contextual and statistical: i.e., they can wait for a given condition to be met before execution, and they support replication over both single test conditions, as well as substitutable elements in the environment. My coauthors and I have used ToyBox to **validate claims made about deep RL agents**. For example, while agents do learn to hit the ball in Breakout at angles and brick configurations that could never have been seen during training, these same agents do not exhibit the higher-level strategy of "tunnelling" claimed in Mnih et al. (2015). Instead, agents appear to follow a fixed pattern of targeting certain bricks in a single column. This example provides **strong evidence of the need for causal evaluation** of deep RL agents.

The PlanAlyzer system also contains logic for evaluation: it **automatically synthesizes contrasts**, or data that may legitimately be compared, for downstream analysis of causal effect in **between-subjects designs**. Synthesizing contrasts only happens after PlanOut scripts have been verified to not contain any internal threats to validity, e.g. selection bias (Tosch et al., 2019a). Future work in this space could synthesize SQL queries or R code that enforce constraints on data analysis.

The final portion of my dissertation is an experimentation platform that **automates experiments** for **explanatory AI**. This work relies on encoding ontologies of concepts, which has connections to the PL concepts of objects and types. Automating sequential decision making will be informed by work in active learning. I am working on a prototype built on top of the ToyBox framework and we expect to deploy a version of this system on Starcraft II as part of an ongoing DARPA-funded project.

4 Future Work

Researchers in PL and SE are increasingly recognizing the need to consider **statistical properties of input data** to programs in order to provide guarantees about such diverse concerns as fairness, privacy, and sensor data (Bornholt et al., 2014; Sampson et al., 2014; Galhotra et al., 2017; Albarghouthi et al., 2017; Near et al., 2019). My work has focused on addressing domain-specific requirements at various points in the data collection and analysis pipeline. I look forward to expanding my current lines of research to include **new data collection instruments and evaluation methods**. Additionally, there are orthogonal research perspectives in other disciplines that I am interested in pursing, with input from domain experts. For example, I look forward to developing my research to encompass questions about **human factors** and **privacy**, which are necessary components to enable responsible and equitable data analysis and experimentation for everyone.

One of the major interim challenges that falls out of my research vision is how to perform **fault localization in machine learning**, due to the emergent properties of of complex models. What is the equivalent of a unit test in a classification system? What is the equivalent of a stack trace in a neural network? Should we expect to find analogues to traditional SE concepts in ML, or does ML demand we rethink software development workflow? How ought we to use techniques such as **delta debugging** or **causal modeling** to identify errors (Zeller, 1999; Chakarov et al., 2016)? I am interested in identifying new debugging methods for ML, especially in the context of iterative computation and experimentation.

Current work on debugging machine learning intersects with efforts to produce **interpretable models** (Doshi-Velez and Kim, 2017). Ideally, an interpretable model is also a composable model; this would allow us to both *understand* and *test* model components. One avenue for developing testable components is to learn or **synthesize probabilistic programs** that represent the data generating process (Nori et al., 2015). Recent work in PL has used probabilistic programming languages to generate low-probability edge cases to improve learning (Fremont et al., 2019). Program synthesis operates in the opposite direction and is one of the most challenging domains for statistical learning techniques, due to discontinuities induced by e.g., conditional branching (Witty and Jensen, 2018). My work will be motivated by the research questions: "How can learned probabilistic programs be composed and reused, and how do we reason about their composition?," and "How can we use this representation to both reason about the properties of the data collection process, and make that process more efficient?"

One avenue for efficiency is in the space of programmatically defined experiments. Current experimentation systems expect to **fetch new data** every time a new experiment is run. The next steps beyond my PlanAlyzer work will involve developing a layer between programmatically defined experiments and the platform for experimentation that can defer this process when needed. For example, I am interested in leveraging work on using techniques from **differential privacy** to bound the **false discovery rate** of inferences made over data collected via various systems (Dwork et al., 2015). This work would also use the techniques I developed during the PlanAlyzer project to ensure the soundness of programmatically defined experiments.

Thus far, my work on DSLs and tool support has focused on finding and encoding appropriate domain-specific abstractions and their compositions for specific tasks. I have not focused on human factors such as usability, visualization, or other cognitive-behavioral perspectives. Moving forward, I would like to collaborate with experts in **visualization and human factors** in computing. For example, a human factors challenge of the PlanAlyzer project was how to best present a large number of valid contrasts to an end-user. At present, it is up to the user to decide the form. Studying how people select treatments from a large number of options could help; data on usability would not only increase the appeal and adoption of my work, but could also inform the direction of my current and future research, e.g. in automated experimentation.

References

Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V Nori. 2017. FairSquare: probabilistic verification of program fairness. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):80.

Eytan Bakshy, Dean Eckles, and Michael S. Bernstein. 2014. Designing and deploying online field experiments. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 283–292, New York, NY, USA. ACM.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279.

James Bornholt, Todd Mytkowicz, and Kathryn S McKinley. 2014. Uncertain<T>: A first-order type for uncertain data. In *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*, volume 49 of *ASPLOS*, pages 51–66. ACM.

Aleksandar Chakarov, Aditya Nori, Sriram Rajamani, Shayak Sen, and Deepak Vijaykeerthy. 2016. Debugging machine learning tasks. *arXiv preprint arXiv:1603.07292*.

Kaleigh Clary, **Emma Tosch**, John Foley, and David Jensen. 2018. Let's Play Again: Variability of Deep Reinforcement Learning Agents in Atari Environments. In *NeurIPS 2018 Workshop on Critiquing and Correcting Trends in Machine Learning*.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. 2015. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638.

Daniel J. Fremont, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. 2019. Scenic: A language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, pages 63–78, New York, NY, USA. ACM.

Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 498–510. ACM.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level Control through Deep Reinforcement Learning. *Nature*, 518:529 EP –.

Joseph P Near, David Darais, Chike Abuah, Tim Stevens, Pranav Gaddamadugu, Lun Wang, Neel Somani, Mu Zhang, Nikhil Sharma, Alex Shan, et al. 2019. Duet: an expressive higher-order language and linear type system for statically enforcing differential privacy. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA):172.

Aditya V Nori, Sherjil Ozair, Sriram K Rajamani, and Deepak Vijaykeerthy. 2015. Efficient synthesis of probabilistic programs. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, volume 50 of *PLDI 2015*, pages 208–217. ACM.

Adrian Sampson, Pavel Panchekha, Todd Mytkowicz, Kathryn S McKinley, Dan Grossman, and Luis Ceze. 2014. Expressing and Verifying Probabilistic Assertions. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, volume 49 of *PLDI*, pages 112–122. ACM.

Lee Spector and Alan Robinson. 2002. Genetic programming and autoconstructive evolution with the push programming language. *Genetic Programming and Evolvable Machines*, 3(1):7–40.

Emma Tosch, Eytan Bakshy, Emery D Berger, David D Jensen, and J Eliot B Moss. 2019a. PlanAlyzer: Assessing Threats to the Validity of Online Experiments. In *Proceedings of the ACM on Programming Languages*, OOPSLA, pages 182–212, New York, NY, USA. ACM.

Emma Tosch and Emery D. Berger. 2014. SurveyMan: Programming and Automatically Debugging Surveys. In *Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications* (*OOPSLA*), pages 197–211, New York, NY, USA. ACM. **Best Paper Award**.

Emma Tosch, Kaleigh Clary, John Foley, and David Jensen. 2019b. Toybox: A Suite of Environments for Experimental Evaluation of Deep Reinforcement Learning. *arXiv preprint arXiv:1905.02825*.

Emma Tosch and Lee Spector. 2012. Achieving COSMOS: A metric for determining when to give up and when to reach for the stars. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 417–424. ACM.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. 2017. Starcraft II: A New Challenge for Reinforcement Learning. *arXiv preprint arXiv:1708.04782*.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2015. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on MachineLearning*, ICML.

Sam Witty and David Jensen. 2018. Causal graphs vs. causal programs: The case of conditional branching. In *Proceedings of the First Conference on Probablistic Programming*.

Sam Witty, Jun Ki Lee, **Emma Tosch**, Akanksha Atrey, Michael Littman, and David Jensen. 2018. Measuring and Characterizing Generalization in Deep Reinforcement Learning. In *NeurIPS 2018 Workshop on Critiquing and Correcting Trends in Machine Learning*.

Andreas Zeller. 1999. Yesterday, my program worked. Today, it does not. Why? In *ACM SIGSOFT Software engineering notes*, volume 24, pages 253–267. Springer-Verlag.