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 previously manual procedural tasks that we traditionally do not think of as computer programs are now either encoded in software, or interact with software. The transition to more automated data collection and decision making carries with it both promise and danger.

For example, 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 programming systems — especially from the fields of programming languages (PL) and software engineering (SE), such as **static analysis** and **code generation** — can be applied to novel tasks in these atypical domains to promote safety and best practices.

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**), **novel runtime systems**, and **testing frameworks** that enforce domain-specific properties of various tasks while aiding in the prevention, diagnosis, and correction of biases in data analysis. While much of my work in is in the *application* of existing tools, techniques, and technologies to novel domains, the requirements that tasks in these domains demand can lead to fundamental research in e.g., programming language design.

The long-term view of my work is to apply programming systems principles in service of developing tools that help **democratize data analysis**, **promote citizen science**, and **facilitate auditing and regulation of complex software systems**. The third of these goals of has recently led me to explore new research problems in **law and PL** — an emerging area within the PL community that leverages recent advances in not only language design and verification, but also machine learning and natural language processing.

Stastistal View		Programming Systems View	
Threat	Instrument	Technique	Toolname/Project
Selection Bias	Field Experiments	DSLs, static analysis, formal verification, code generation	PlanAlyzer (Tosch et al. (2019a); Clary et al. (2022)), **CACM Research Highlight**
Measurement Bias Construct Validity	Surveys	DSLs, static analysis, dynamic analysis, information flow	SurveyMan (Tosch and Berger (2014), **Distinguished Paper Award**)
Confounding	Simulation	software testing, dynamic analysis	ToyBox (Tosch et al. (2019b), Clary et al. (2018))

Program analysis is a *method* for combating threats to the internal validity of data collection instruments.

Dynamic Data Collection Instruments as Programs

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., exported application logging).

Much of my research thus far has focused 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. I designed the DSL as a **spreadsheet-based language** that would integrate with researchers' existing practices. Because the DSL included branching and random selection, execution was nondeterministic, leading us to employ both static and dynamic analyses to verify and monitor correctness. This work won first place at the **Student Research Competition** at PLDI 2014, and a **Best Paper Award** at OOPSLA 2014.

Ongoing and Future Work. I have recently returned to this work due to both interest from a student collaborator and the evolution of machine-learned coauthorship tools. At UVM I began working with a student on applying **quantitative information flow** to **programmatically-defined adaptive surveys** in service of both security and privacy concerns. In the future I expect to use ML tools to improve the usability and adoption of this work.

Security Concerns. Paid online surveys suffer from data integrity issues due to the threat of bots and low-engagement participants. The existing mechanism in SurveyMan for detecting adversarial behavior uses clustering and an entropy-based metric for flagging low frequency responses. A major shortcoming of this threat model is that it cannot be used in contexts where low frequency responses are critically important, since the rejection rate would be too high. A future work goal is to develop a schedule of questions and associated pricing model that would compensate participants commensurate with the value added to the survey data, while excluding data that are truly from adversarial respondents.

Privacy Concerns. Respondents who give low frequency responses may be concerned about privacy violations through e.g., linking attacks. Thus we began looking into the integration of measures of information content in text responses and their possible correlations with other questions: text response questions are often reported in full or part, but can leak information about the respondent. As preliminary work, we investigated stylometric analyses and dialect classification, with obfuscation as a potential remedy to this leakage.

Usability Concerns. The original SurveyMan DSL makes for a suitable intermediate representation; as an end-user language, it is too idiosyncratic. Furthermore, it requires effort on the part of the survey author to generate novel question wording. Thus I am interested in working on co-authorship of surveys with ML tools, including machine translators and GPT-3, and integrating these tools into a spreadsheet program. Once I have an interested student, I intend to pursue a collaboration with an interested colleague at Microsoft Research on using surveys to study textual entailment.

Experimental Design as a Programming Task

OOPSLA 2019, USENIX Security 2022; SIGPLAN Research Highlight 2020, CACM Research Highlight 2021; NSF FMitF-2220422 "FMitF Track I: Formal Methods in Software Support for Sound Experimentation", $661,021 (2022-2026)

Experimentation increasingly drives the decisions in digital or online public spaces. Large firms have developed sophisticated experimentation management systems, including software frameworks and domain-specific languages (DSLs) for designing, writing, deploying, and analyzing experiments at scale. Unfortunately, this work happens in walled gardens, away from the public eye. Researchers seeking answers to questions about human behavior in such systems must either settle for observational data released by the firms that operate the infrastructure, acquire permission to run experiments at the firm itself, or run small-scale experiments on similar, independently operated infrastructure. There are critical limitations to each of these arrangements: some effects of interest may not be identifiable from observational data alone, running experiments at private for-profit firms presents a host of both logistical and ethical issues, and smaller-scale replicas of such infrastructure may not have external validity.

While the cost of running an "incorrect" experiment in most cases amounts to wasted resources, incorrect conclusions or misallocation to experimental treatments can be a public relations disaster: it incentivizes the firm to restrict information while potentially causing a backlash against experimentation in such systems. This is bad for science and bad for policy, since it can turn the public against experimentation and cause misunderstandings about the purpose and necessity of experimentation in such systems.

One major challenge that undercuts these issues is that such socio-technical systems were not designed to allow for experimentation. As new domains and platforms develop, the underlying infrastructure ought to support experimentation best practices, both for scientific endeavours and to aid in transparency for regulatory purposes. Unfortunately, most of the existing infrastructure work has happened within for-profit companies, which may not release reproducible artifacts or open-source software. There are few incentives for firms to enable the competition to better experiment. Most critically, however, there is very little public work available on enabling verifiably sound experimentation in socio-technical systems, and it is unclear whether any formal methods have been applied in this space. My work on PlanAlyzer — a static analyzer for programmatically-defined experiments that focused on combating selection bias — is the first of its kind in its attempt to address issues in this space (Tosch et al., 2019a).

Ongoing and Future Work. I am currently working on the first non-industrial attempt to formalize the experimentation-analysis pipeline in socio-technical systems. This pipeline ties together hypothesis registration, treatment allocation, and downstream statistical analysis, tightly coupling each phase of the pipeline in software. The first phase of this work — a chimeric language called Helical consisting of two sub-languages that capture hypotheses and treatment assignment — is near completion. I have simultaneously been collaborating with a colleague in database theory to develop a description logic to constrain the set of permissible queries with respect to a fixed database schema that is populated from Helical programs. I expect to employ Helical in the classroom when I next teach my *Systems for Knowledge Discovery* course, which doubles as a research methods course.

```
O : { "02", "03" }
Y : nat
(progid) Y <- O
sharp (progid) assert (Y > 0)
Y_A = Y | O = "02"
Y_B = Y | O = "03"
(progid) assert (Y_A > Y_B)
```

Figure 1: Partial Helical program.

I intend to prove that Helical programs have identifiable effects, automatically generating estimators for end-users that are consistent with the registered hypotheses. Building upon existing work in gradual typing and embedded languages, I will then show how Helical can be integrated into existing codebases and languages, obviating the need for a standalone DSL, which will increase the usability and likelihood

of adoption.

Unfortunately there are no publicly available corpora for experiments. Thus, to evaluate the validity of this formalism and spur research in this emerging area, I am building a corpus of experiments drawn from "found" experiments on the web, published papers, and new experiments directly encoded in my novel formalism. Corpus building and empirical analysis of current practice in software-mediated experiments is critical to the success of this work. I intend to submit the results of my corpus-building and empirical analyses to software engineering and data science conferences (e.g., ICSE, FSE, CODE, or KDD).

In the course of developing this work, I have encountered great interest from the cybersecurity community via researchers who focus on promoting reusable artifacts and replication and reproduction of cybersecurity findings. Lifting experiments to a formal language has additional benefits beyond the immediate verification and code generation I have enumerated. Formal structural representations of experiments can aid in searching for prior work, flagging work that requires replication, and correcting past mistakes. While such capabilities ought to interest a broad array of research communitites, the cybersecurity community's focus on methods and measurement make it a particularly appealing target for the adoption of my work.

Software Testing for Learned Software

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**. 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).

In some domains, researchers have more control over the platform in which they do experiments. One such domain where platform design deserves a more principled look is in the evaluation and **explanation of 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. ToyBox has already influenced, and is being used in, several machine learning research projects in my former group. Furthermore, an early ToyBox prototype I designed and developed has been used by our collaborators 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 (Witty et al., 2021). ToyBox also includes a **behavioral testing framework**; this framework would not be possible without platform support for intervention.

Ongoing Work. 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.

My research has also taken steps toward **automated experiments** for **explanatory AI** (Tosch, 2020). This work relies on encoding ontologies of concepts, which has connections to the PL concepts of objects and types. I found that explaining behavior in **relational and time-varying environments** required developing an explanation typology (random, trivial, frame-violating, and causal) and posited that only one explanation type would be satisfactory to end-users: those that were causal. I built a prototype of this explanation system on top of the ToyBox framework and have been working to apply it to other simulation environments and to evaluate my hypotheses with human subjects. There is a direct line from early work I did on stopping conditions for evolutionary algorithms to this automated experimentation work Tosch and Spector (2012).

Programming Languages and Law (PL+Law)

My focus on data collection has led me to fortify my background in statistics and causal inference. The literature on **causal inference** differentiates *effects of causes* and *causes of effects*. The former describes experimentation, but the latter describes explanation and many of the examples in the literature are concerned with legal reasoning. This background, combined with recent trends in the PL community, has motivated me to expand my research interests into **computational law**, especially with respect to **PL and formal methods**.

Long-term, I would like to explore the connections between experimentation and law; short-term, I have worked with several students on preliminary projects in the area of **PL+Law**. Over summer 2022, I worked with a recent UVM graduate (currently a *1L* at University of Wisconsin Law School) in the study of informed consent for participation in experiments vis à vis terms of service agreements and modal logic. We were specifically looking at the role of experimentation and whether it was covered by contracts of adhesion. We plan to submit our work to FAccT 2023.

I had previously begun working with a student on formalizing two possible domains: insurance contracts and arbitration agreements. We were inspired by Basu et al. (2019), which formalized the transfer of interest in property as a programming language. Our plan was to use the formalization process as a means to generate scenarios that are entailed by these legal documents, and use those scenarios test end-user understanding of the implications of agreeing to these contracts. This student was unable to continue on this work due to visa issues, but I have been able to pursue this work through my postdoctoral researcher position at Northeastern under Dr. Chris Martens, where I was able to join existing efforts in this domain.

The project I have joined uses **narrative generation** to explain the consequences of policies to end-users; the lead graduate student on this project will be presenting at the *Workshop on Programming Languages and the Law* in January (Dabral, Tosch, and Martens, 2023). Narratives are generated using **answer set programming (ASP)**, an alternative to traditional SMT-based solvers. ASP allows for greater flexibility for knowledge representation, due to its relationship with **default logics** and its ability to easily capture the possible world semantics of modalities. I expect to apply ASP in my other lines of research, due to its flexibility. I also expect to continue this work at my next location and am particularly interested in building ties with a local law and/or policy school (e.g., Schar School of Policy and Government or Antonin Scalia Law School).

Concluding Remarks

I am primarily motivated by research questions that address the integrity and reliability of data collection in software. While the primary methods I use come from PL and SE, there are many related research questions that require additional expertise. Thus far I have sought out that expertise myself, but I look forward to working in a larger research environment where I can more readily find collaborators; my work has broad applications and presents many opportunities for interdisciplinary collaborations.

References

Shrutarshi Basu, Nate Foster, and James Grimmelmann. 2019. Property conveyances as a programming language. In *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, pages 128–142.

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.

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*.

Kaleigh Clary, **Emma Tosch**, Jeremiah Onaolapo, and David Jensen. 2022. Stick It to The Man: Correcting for non-cooperative behavior of subjects in experiments on social networks. In *31st {USENIX} Security Symposium*.

Chinmaya Dabral, **Emma Tosch**, and Chris Martens. 2023. Exploring consequences of privacy policies with narrative generation via answer set programming. *Workshop on Programming Languages and the Law*.

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 –.

Emma Tosch. 2020. *System Design for Digital Experimentation and Explanation Generation*. Ph.D. thesis, University of Massachusetts.

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. **CACM Research Highlight**.

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.

Sam Witty, Jun K Lee, **Emma Tosch**, Akanksha Atrey, Kaleigh Clary, Michael L Littman, and David Jensen. 2021. Measuring and characterizing generalization in deep reinforcement learning. *Applied AI Letters*, 2(4):e45.