Lucas Saldyt

MIT Statement of Purpose

I am interested in working with Dr. Armando Solar-Lezama, Dr. Omar Reyes, Dr. Michael Carbin, Dr. Leslie Kaelbling, Dr. Tomás Lozano-Pérez, or Dr. Joshua Tenenbaum. Ideally, I would like to research neurosymbolic program induction as a model for cognition and robotics, specifically learning, world modeling, and planning. Neurosymbolic programming has emerged as a leading paradigm for artificial intelligence and cognitive science. Neurosymbolic programming is highly interdisciplinary, which makes MIT a perfect fit because of its eclectic collection of brilliant professors, research scientists, and students. For example, because Dr. Solar-Lezama, Dr. Omar Reyes and Dr' Carbin's expertise is in program synthesis/analysis, Kaelbling's is in planning and reinforcement learning, Lozano-Pérez is in robotics, and Tenenbaum's is in cognitive science, together their combined knowledge is ideal for studying neurosymbolic programming as a model for cognition and robotics. In my research, I define robots as embodied computers that act through composite computer programs connecting sensation to actuation. Similarly, computational cognitive science models human intelligence as an advanced computer program, and in a sense, humans are also embodied computers! Thus, neurosymbolic programming is a natural fit for robotics and cognitive science. For example, program synthesis algorithms can induce options for hierarchical reinforcement learning or induce world models, such as intuitive physics/psychology, logical rules, or PDDL (Planning Domain Description Language). Overall, I envision neurosymbolic program induction as integral to a broader cognitive architecture that models human intelligence.

Since July, I have been working on an idea called "synthesized differentiable programs," which combines program synthesis and neural networks through neural compilation. This idea is similar to probabilistic programming and program synthesis through sketching. In my proposed algorithm, a program synthesis algorithm generates template programs, and then neural compilation allows for syntactic computer programs to be compiled into the weights of a differentiable network. I implemented the proof of concept for this idea, which was accepted into the NeurIPS 2022 Neural, Causal and Symbolic AI workshop. I plan to extend this work and use this technique as a basis for more specialized neurosymbolic programming tasks. For instance, neural compilation can compile expert planning or combinatorial optimization algorithms (which I've also explored in a class project using SATNet for interpretable differentiable planning). Effective neural compilation can also efficiently represent procedural knowledge. I anticipate many follow-ups, such as creating flexible/scalable neural compilation algorithms (e.g., for the Transformer architecture) or increasing interpretability through sparsity constraints.

Neurosymbolic programming has many applications in reinforcement learning and planning. For example, DreamCoder finds hierarchical options/macros by inducing an abstract domain-specific language via program synthesis. If it were connected to low-level policies, it would be possible to do neurosymbolic task and motion planning, such as in "Hierarchical Task and Motion Planning in the Now," or, naturally, "Learning Neuro-Symbolic Skills for Bilevel Planning." Another exciting approach is "Learning to synthesize programs as policies," which learns both an explicit program and a latent policy through anchoring. In July, I proposed these ideas to Dr. Kevin Ellis, who suggested the possibility of inducing world models as neurosymbolic programs, which I'm pursuing as a project in Dr. Subbarao "Rao" Kambhampati's class on planning and learning. This algorithm learns multi-resolution world models by filling in a human-expert template via program synthesis and neural anchoring. I'm interested in how well acquired symbols (induced programs) correspond to human-vocabulary symbols, for instance, if a compression/complexity objective is effective at re-creating human abstractions. I'll submit this to IJCAI 2023 and extend it over the next year. I'm also considering how world model induction relates to the Konidaris/Kaelbling/Lozano-Pérez notion of symbols as classifiers, as proposed in "Constructing (learning) symbolic representations for high-level planning," and "From skills to symbols."

My first-year Ph.D. research focused on Turing-complete long-horizon robotics tasks, which I modeled as differentiable program induction using various differentiable computers and memory-augmented transformers. Robot behavior needs to be Turing-complete for many complex long-horizon robotics tasks involving memory, search, planning, and world modeling. Also, memory is required to learn how to augment non-Markovian state information and reason in belief space (which is necessary to handle partial observability or partial goal specification). In this research, I formulated the "Hybrid Traveling Salesman Problem," where a robot in a continuous space must optimally traverse through goal configurations. I then proved that solving this problem optimally would require coupling since the dynamics of a robot constrain feasible solutions. I also formulated this problem using the differential method of Lagrange multipliers to account for feasibility, interpretability, and safety constraints. Initially, I approached this problem using entirely neural methods, which works superficially but is ultimately limited since, for instance, state-of-the-art neural approaches can, at best, solve traveling salesman instances with 100 cities. In comparison, classical solvers can scale to 85,900 cities. Accordingly, in my current neurosymbolic reasoning class, my teammates and I created a transferable and interpretable differentiable combinatorial solver based on the mixing method for MAXSAT/SDP. Differentiable SAT solving overcomes the limitations of purely neural approaches but retains the flexibility of differentiable function approximation. Overall, this early Ph.D. research inspired my later work on neural compilation, differentiable combinatorial solvers, and world model induction. My future ambition is to use these components to create a general neurosymbolic cognitive architecture.

My final-year undergraduate research focused on fine-tuning robot path planners to specific environments. Specializing in a particular environment can confer, on average, a 70% reduction in computation and a 10% reduction in path lengths. I modeled this with a genetic program synthesis algorithm called Pareto evolution, which would evolve Python code for sampling-based path planners. By re-arranging sampling code and optimizing hyperparameters, it was possible to find a Pareto-front including both generally effective planners and highly effective planners fulfilling particular niches like efficiency, exploration, or optimality. Furthermore, the learned specializations are highly human-interpretable since the end result is simplified Python code. This approach was applied to hundreds of video-game maps, such as StarCraft, and thousands of real-life city maps. We also explored the possibility of prioritizing exploration efficiency, which was effective for pre-computing planning graphs to use on specific maps. The original paper was published in the ICLR 2021 "Learning to Learn" workshop, and the extension was published in the ICRA 2021 "Curious Robots" workshop (which Dr. Tenenbaum and Dr. Lozano-Pérez both spoke at).

I knew I wanted to pursue a Ph.D. for at least eight years, but I only learned what topics I wanted to research a year into my Ph.D. after reading 1,135 papers. After eight years of exploration, I decided to pursue neurosymbolic programming at MIT. Mor Harchol-Balter points out that a Ph.D. is about doing "whatever it takes" to solve a research problem: "(taking five math classes, learning a whole new area like databases, rewriting the whole kernel..)," and I hope to show that I have this dedication. In my case, I am transferring universities to pursue a research problem that I consider to be at the heart of modern artificial intelligence: neurosymbolic programming for robotics. Arizona State University offers phenomenal opportunities, and I am thankful for my time there, especially for exposure to world-class professors like Subbarao Kambhampati and Dmitri Bertsekas. I owe doctors Kambhampati, Shakarian, Pavlic, Colbourn, and Amor dearly for their insights and mentorship. As a point of fact, ASU has neither a course nor a professor focused on program synthesis. Alternatively, MIT has specialized faculty in program synthesis, robotics, planning, reinforcement learning, and cognitive science, and it is a perfect fit for my research.

My decision to pursue program synthesis research didn't come lightly. It began eight years ago, in high school, when I became obsessed with programming languages. I completed a senior project where I created a language-to-language compiler that supported subsets of Python, C++, Haskell, and Fortran. In doing so, I learned about parsing, grammars, the Chomsky hierarchy, and the difficulties, nuances, and niches of each language. This knowledge became invaluable to my eventual research, where I work with grammars and languages daily.

The best mentor I learned from was Dr. Erik Nielsen at Sandia National Labs. Erik was an undeniable genius, with four large crates of quantum electrodynamics notes from his graduate study at Princeton. He had frequent meetings and responsibilities with the DOE, but even so, he would take the time to give me, a high schooler, lessons in basic quantum computation. He was a talented software engineer who would help me debug even the most minor Python bugs. Judging by the success of their students, Dr. Tenenbaum, Kaelbling, Solar-Lezama, Carbin, and Lozano-Pérez are all clearly excellent mentors. Brendan Lake is now a world leader in cognitive science, Kevin Ellis has become a world leader in program synthesis, and George Konidaris is now a world leader in hierarchical reinforcement learning. I aspire to mentor my future students with the same effort and care.

As a sophomore at ASU, I used my Sandia Labs internship knowledge to create a small functional quantum programming language called Qurry. I presented this at the FOSDEM quantum software workshop in Brussels, Belgium. Later, I gained my first exposure to Joshua Tenenbaum and the idea that programming languages can explain human cognition (the child as a hacker). I also took my first forays into algorithmic information theory, which I pursued in the context of a symbolic regression project, but led me to understand the heart of Francois Cholet's intelligence measure, Marcus Hutter's "Universal Intelligence," and DreamCoder.

My first-year Ph.D. research started with a literature review phase, notably including Dr. Solar-Lezama's course on program synthesis and DreamCoder. However, my advisor and I decided to focus on differentiable program induction, closer to his expertise. I initially replicated the "Neural Turing Machine" and "Differentiable Hybrid Computing" papers from scratch. These approaches are mathematically brittle, and I spent months making them less so. Luckily, their shortcomings influenced my research: Differentiable computing is based on the hypothesis that a supervised training signal is sufficient to uncover desirable programs from random initialization. This relies on the theoretical fact that the weights of a neural network can represent desirable programs. *When I thought about this deeply, I re-discovered the possibility of neural compilation.* To truly dig into program induction, I took the most challenging theoretical computer science course my program offered and earned the highest grade that semester. I used this background to design an independent study course digging into fundamental papers relating to Turing completeness and neural networks, such as the original Siegelmann and Sontag or Gruau's 1995 neural compiler, as well as more modern papers. Finally, I related neural compilation back to program synthesis and implemented the "synthesized differentiable programs" paper in September, which I wrote independently.

In the upcoming semester, I'll intern at NASA JPL to deploy advanced planning algorithms on the Mars Perseverance Rover and Ingenuity helicopters. I am using the opportunity to learn more about the real-world constraints that affect machine learning algorithms and direct my Ph.D. research toward creating formal verification and learning algorithms that can be deployed in safety-critical scenarios, especially robotic space exploration.

My primary career goal is to become a university professor, where I plan to become a leader in the field and inspire the next generation of students. Over the previous summer, I spent hundreds of hours tutoring diverse and non-traditional students in math and computer science. Seeing students succeed is incredibly rewarding, especially when typical classroom instruction has let them down. The difficulty of teaching is highly underestimated, not only by new teachers but also experienced professors. However, the benefit of education is exponential: many of the students in a classroom of a hundred will, in turn, teach hundreds of others, and thus teaching is criminally undervalued compared to research, especially since generations of students will be researchers themselves. This fact is particularly true for students who have been dissuaded for any reason, and it is my duty to prevent this. A teacher's support can make or break a student's career, and many students who avoided STEM can point to a single class that dissuaded them. While tutoring, I read through the most popular math and computer science textbooks and designed a unique study plan for each student. I regularly encouraged my students, especially those from diverse backgrounds, to apply for opportunities such as NASA internships. I would also edit their resumes and help them prepare for coding interviews. Notably, I helped a friend from a historically marginalized background prepare his application materials for NASA Langley, where he will be starting full-time as a computer vision researcher. At MIT and in my future career, I will put in the hours necessary to inspire the next generation of students to pursue their dream opportunities regardless of their background.

I plan to pursue a lifelong career in scientific research, and I'm motivated by curiosity, novel scientific discovery, and the possibility of applying research to cutting-edge engineering. Neurosymbolic programming is a leading model for not only creating artificial intelligence but also for understanding natural intelligence. I hope to extend this theory substantially through dedicated, diligent work. Also, there is incredible potential for neurosymbolic programming in engineering, such as space robotics. Completing a Ph.D. at MIT will allow me to contribute to a broader neurosymbolic cognitive architecture and model for human intelligence. My research will not end with my Ph.D. In the future, I hope to lead a team of researchers toward scientifically understanding intelligence in both humans and robots.

— Lucas Saldyt, an aspiring scientist hoping to join MIT