Research Statement

Tom Silver

My research objective is to develop robots that **learn to plan** and **plan to learn**. To build broadly competent intelligent robots—ones that work in homes, factories, hospitals, restaurants, stores, assisted living facilities, and disaster zones—we need planning and learning, and they need each other. We need **planning** so that a robot given a command like "wipe down the inside of the refrigerator" can reason about constraints in the moment: move the milk out of the way, but leave it in the fridge (so it doesn't spoil) and not on top of the eggs (they'll crack). We need **rapid, generalizing, abstract learning** so that the robot understands from one demonstration that a plunger can be used for plunging, and also as an improvised tool for sweeping out-of-reach objects; and **slower, specializing, low-level learning** to perfect skills like sweeping or concepts like "suction". We need to integrate learning and planning so that the robot can acquire new skills and concepts, compose them together with existing ones, and collect experience to improve future decision-making. In this virtuous learning-planning cycle, learning serves planning *and* planning serves learning (Figure 1).

Towards realizing this vision, my research considers **learning and planning with abstractions** using techniques from task and motion planning (TAMP), program synthesis, reinforcement learning, and neuro-symbolic learning. I developed the first unified system for learning abstractions for TAMP [1 -- best paper finalist at IROS 2021; 2-4], advanced the state-of-the-art for planning in very large problems by learning to attend to relevant objects [5, 6 -- oral at CoRL 2020], solved even larger problems by synthesizing programmatic policies from very few examples [7-10], and implemented a virtuous cycle for abstraction learning [11, 12 -- oral at CoLLAs 2023]. A guiding principle in my work is that learning and planning should be *pragmatic*[1]: models should be trained to make good decisions quickly. This commitment leads to a middle ground between "pure planning", which makes good decisions slowly, and "pure learning", which makes bad decisions quickly when data is scarce.

The following sections describe my progress towards this middle ground and my long-term vision for **pragmatic robot planning and learning**. Directions that I plan to pursue next include: (1) generalizing and transferring abstractions between radically different domains; (2) leveraging foundation models for abstraction learning; (3) applying these techniques to caregiving robotics; and (4) democratizing robot planning and learning through open-source software. These advances will lay the foundation for a cognitive architecture that will power the broadly competent intelligent robots of tomorrow.

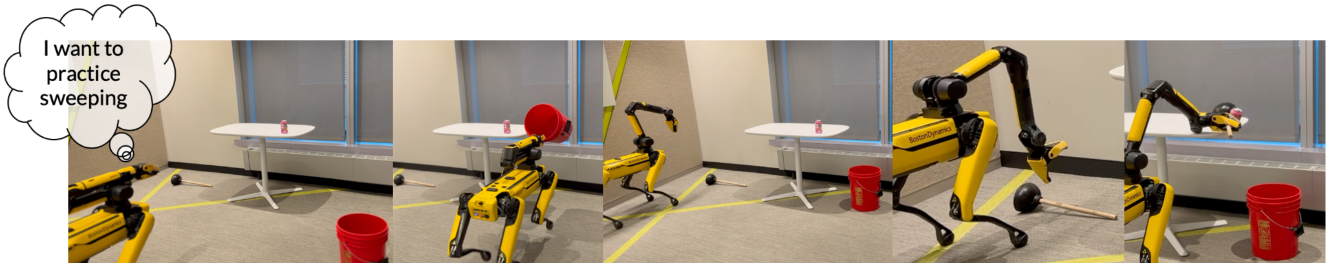


Figure 1: Planning to learn. The robot decides what to practice (sweeping with a plunger) and how to practice it (continuous parameters). It then uses planning to autonomously set up a scene where sweeping is possible.

[1] In the sense of William James (1906): "Our beliefs are really rules for action... to develop a thought's meaning, we need only determine what conduct it is fitted to produce: that conduct is for us its sole significance."

Previous and Current Work

Learning Abstractions for Planning. When robots are embedded in everyday life, they will need to make very long-term plans. One strategy for long-term planning is to decompose decision-making into a high level ("what to do") and a low level ("how to do it") (Figure 2). Reasoning at the high level requires **state abstractions**, like "the cup is in the back of the cabinet", and **action abstractions**, like "push the cup to the back". It also requires a mechanism to **ground** abstract states in sensory inputs and another mechanism to ground abstract actions in motor actions. Traditionally, the state abstractions, action abstractions, and grounding mechanisms are all designed by hand on a per-domain basis. This manual design is time-consuming and difficult, even for domain-expert programmers.

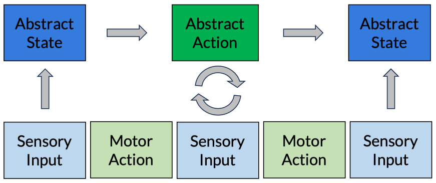


Figure 2: Planning with abstractions

Task and motion planning (TAMP) is a form of planning with abstractions that builds on decades of progress in classical AI planning, which addresses "what to do", and continuous optimization, which addresses "how to do it". In my PhD work, **I developed the first unified system for learning all of the abstractions and grounding mechanisms needed to do TAMP** [1-4]. The abstractions are explicitly learned to make *good decisions quickly*, and they often outperform human-designed abstractions in terms of planning time and success rate.

A central challenge addressed by my work is that abstractions are necessarily *lossy* in complex environments. For example, if the robot is holding a hammer, can it place it on a nearby table? Given an abstract action `Place(hammer, table)` with a precondition `Holding(hammer)`, the answer may appear to be yes, but the real answer depends on the morphology and kinematics of the robot, the geometry of the table and other objects, and the grasp of the hammer. It is tempting to learn abstractions that are as lossless as possible, but this is often hopeless (e.g., we would need a predicate like `HammerOnTableNotObstructingWrenchPlacement(hammer, table, wrench)` to fully model placing both objects on a small table—this would certainly not scale to complex environments with many interacting objects). The pragmatic view suggests a different path forward: we should learn abstractions that are **explicitly optimized for efficient and effective planning**. For example, in my predicate learning work [3], I derived a surrogate objective that analytically approximates planning time from demonstration data. I then borrowed techniques from program synthesis to learn predicates that optimize this objective. The learned predicates are interpretable, but sometimes surprising in their planning benefits. For example, in the classic blocks world domain, the planner exploits the learned predicates to achieve a 30x time reduction over the standard encoding.

Learning To Accelerate Planning. Even with good abstractions, state-of-the-art planners are slow when tasks feature hundreds of objects. This is especially limiting considering that the notion of an "object" is flexible—another abstraction that the robot creates—and may include, for example, "the back left leg of the chair". To accelerate planning, I developed techniques that use graph neural networks [5] and probabilistic graphical models [6] to **learn to attend to relevant objects**. In [6], the robot learns to impose constraints on itself so that more objects can be ignored. For example, the robot learns to top-grasp objects in clutter and re-grasp them before placing them onto a shelf. Such a plan has more steps than if the robot carefully motion-planned around the clutter to side-grasp the objects, but the overall *planning time* is substantially shorter, meeting our pragmatic objective.

When many objects are relevant, we need additional mechanisms to accelerate decision-making. One strategy is to circumvent slow, deliberative abstract planning by **learning an *abstract policy* that quickly proposes actions** [7-10]. These abstract policies can be implemented as programs and learned using program synthesis techniques. For example, in the "Reach for the Star" domain (Figure 3), a general "build stairs and climb" program, with for-loops and conditional statements, can be learned from 3 examples and generalized to build arbitrarily large stairs [8]. I have shown how inductive logic programming [7], Bayesian decision-tree learning [8], SAT solvers [9], and large language models [10] can be leveraged to learn complex abstract policies **involving >100 subprograms from 5 or fewer examples**.

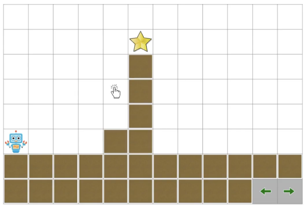


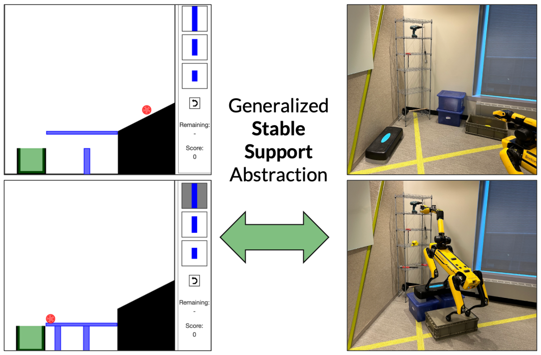
Figure 3: Reach for the Star

<u>Planning to Learn.</u> Abstract states, actions, and policies can and should be learned from very few examples, but *grounding* the abstractions in sensory inputs and motor actions takes practice—active learning in the real world. Furthermore, **active learning requires planning**: the robot needs to decide *what* data to collect and *how* to collect it by planning ahead. I have considered planning-to-learn for abstract action learning [11] and abstract state grounding [12]. In the latter work, my Master's student Amber Li and I considered a setting in which a teacher is situated with the robot in the environment. The robot is permitted to ask the teacher about abstract state groundings in the current environment ("Is `On(fork, plate)` true?") and take actions to change the environment. Importantly, since the robot can only query the teacher about the current state, it must plan to reach states where "interesting" questions can be asked. In ongoing work at the Boston Dynamics AI Institute, I am enabling a Spot robot to plan-to-learn (Figure 1). Given a set of imperfect skills, the robot repeatedly chooses a skill and plans to practice it. The robot can collect data for hours without intervention, re-planning to compensate for noise. **Through practice, the robot learns to get better at planning, which in turn supports planning to learn, completing a virtuous cycle**.

Future Directions

The work described thus far represents the beginning of a long-term research program in pragmatic robot planning and learning. The future directions that I will prioritize include:

<u>Refactoring, Generalizing, and Transferring Learned Abstractions.</u> I want to enable robots to learn and plan with very general versions of *physical* abstractions: containers, trays, levers, stable supports, dials, bridges, and many others. An intelligent agent should be able to "create a stable support", for example, in 2D physics puzzles *and* real-world scenarios (Figure 4). Kelsey Allen, the co-creator of the Virtual Tools Game (left), applied our programmatic policy learning approach [8] to the benchmark and found promising initial results. But to create **truly general abstractions that transfer between radically different domains**, like leveraging the concept of "stable support" to build a **makeshift staircase (right)**, we will need *refactoring* from the program synthesis literature, a direction I am exploring in ongoing work with Prof. Sebastijan Dumancic at TU Delft.



(Allen et al. 2020)

"Fetch the drill"

Figure 4: Planning with a "stable support" abstraction

Learning and Planning with Foundation Models. Large language models (LLMs) and vision-language models (VLMs) have multiple use cases in my research agenda. In the short term, I have already found that LLMs can reliably translate natural language commands into symbolic goals for planning, and that VLMs can be used for object detection with Spot (Figure 1). I have also proposed methods for improving planning [14] and abstract policy learning [10] using LLMs. In the medium term, I want to **use both LLMs and VLMs to improve state and action abstraction learning**. VLMs in particular have some ability to ground natural language concepts like `SuctionAttached(plunger, surface)`; we should leverage this knowledge to bootstrap abstraction learning. In the long term, I would like to explore natural language as a substrate for communicating between many (many!) different modules in a **large, integrated cognitive architecture**.

Assistive Robotics Applications. My mother passed away in January 2020 after battling a rare blood disorder. In the last few months of her life, her mobility and independence were significantly reduced, which put an enormous strain on her mental health. This experience left me determined to direct my work towards applications that would assist people in similar situations. From conversations with robotic caregiving experts, I have come to understand that **there is tremendous potential to leverage planning, especially TAMP, in assistive applications**. Instrumental activities of daily living (IADLs) like cleaning, object retrieval and storage, and meal preparation offer clear opportunities, but activities of daily living (ADLs) like assistive feeding, dressing, and grooming also require decision making with multiple levels of abstraction. While integrating minimally-intrusive robots into existing homes should be our ultimate goal, we should also consider what can be achieved through **physical design coupled with robot planning**. Drawers can be made actuatable; refrigerators can be equipped with RFID readers; and planning can treat an entire smart home as one large robotic system.

Democratizing TAMP Learning. I am frequently approached by other researchers who want to use TAMP learning in their projects. I do my best to help them get started, but it's clear that **the barrier to entry for TAMP learning remains far too high**. The field needs a leader to prioritize accessibility, and I am well-positioned to be that leader. My PDDLGym [14] library, designed to make task planning more accessible to people familiar with reinforcement learning (RL), is the most popular PDDL-language repository on GitHub[2]. My 2021 survey paper with colleagues has quickly become the de facto reference for TAMP [13]. I have organized three workshops at CoRL and RSS on TAMP learning and related topics. Finally, my research on learning abstractions has long been motivated by the prospect of making TAMP a drop-in replacement for reinforcement learning (RL). In the same way that one can run RL algorithms on environments that conform to a standard API, I want to enable people to **`pip install tamp-learning`** and run our algorithms with almost no setup.

Looking Further. My very long-term goal is to develop a full cognitive architecture for general-purpose robots. Achieving this goal will require deep collaborations and a commitment to cumulative and coherent research—the architecture components need to work well *together*. Pragmatism will be a useful guiding principle throughout this multi-decade project. We want robots that make good decisions quickly in homes, factories, hospitals, restaurants, stores, assisted living facilities, and disaster zones, so we should train them that way.

[2] Popularity based on repository star count as of October 2023.

References

1. **Tom Silver***, Rohan Chitnis*, Joshua Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning symbolic operators for task and motion planning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.
2. Rohan Chitnis*, **Tom Silver***, Joshua Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.
3. **Tom Silver***, Rohan Chitnis*, Nishanth Kumar, Willie McClinton, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Joshua Tenenbaum. Predicate invention for bilevel planning. *AAAI Conference on Artificial Intelligence (AAAI)*, 2023.
4. **Tom Silver,** Ashay Athalye, Joshua Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning neuro-symbolic skills for bilevel planning. *Conference on Robot Learning (CoRL)*, 2022.
5. **Tom Silver*,** Rohan Chitnis*, Aidan Curtis, Joshua Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Planning with learned object importance in large problem instances using graph neural networks. *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.
6. Rohan Chitnis*, **Tom Silver***, Beomjoon Kim, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. CAMPs: Learning context-specific abstractions for efficient planning in factored MDPs. *Conference on Robot Learning (CoRL)*, 2020.
7. Ryan Yang*, **Tom Silver***, Aidan Curtis, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PG3: Policy-guided planning for generalized policy generation. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.
8. **Tom Silver**, Kelsey R. Allen, Alex K. Lew, Leslie Pack Kaelbling, and Joshua Tenenbaum. Few- shot Bayesian imitation learning with logical program policies. *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.
9. Aidan Curtis, **Tom Silver,** Joshua Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Discovering state and action abstractions for generalized task and motion planning. *AAAI Conference on Artificial Intelligence (AAAI)*, 2022.
10. **Tom Silver**, Soham Dan, Kavitha Srinivas, Joshua Tenenbaum, Leslie Pack Kaelbling, Michael Katz. Generalized Planning in PDDL Domains with Pretrained LLMs. *Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL-IJCAI)*, *2023.*
11. Rohan Chitnis*, **Tom Silver***, Joshua Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. GLIB: Efficient exploration for relational model-based reinforcement learning via goal-literal babbling. *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.
12. Amber Li, **Tom Silver**. Embodied active learning of relational state abstractions for bilevel planning. *Conference on Lifelong Learning Agents (CoLLAs)*, 2023.
13. Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, **Tom Silver**, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.
14. **Tom Silver***, Varun Hariprasad*, Reece S Shuttleworth*, Nishanth Kumar, Tomás Lozano-Pérez, Leslie Pack Kaelbling. PDDL planning with pretrained large language models. *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.