Learning By and For Task and Motion Planning: A Survey

Yixuan Huang[1,*], Benned Hedegaard[2,*], Naman Shah[3], Siddharth Srivastava[4], George Konidaris[2], and Tom Silver[1]

Abstract

Task and motion planning (TAMP) enables robots to reason jointly about "what to do" (task planning) and "how to do it" (motion planning), building on decades of advances in classical AI planning and continuous optimization. Despite this progress, traditional TAMP methods remain limited by engineering overhead and planning speed. A growing body of recent work has aimed to address these limitations by combining TAMP with machine learning. Yet these efforts remain fragmented across different formulations, assumptions, and evaluation settings, making it difficult to assess progress or identify core challenges. We present the first unified survey of this combination, organizing the area around three families of approaches. The first two leverage learning to mitigate TAMP's bottlenecks: learning to make TAMP possible (to mitigate engineering overhead) and learning to make TAMP fast (to accelerate planning). The third inverts the direction: using TAMP for learning repurposes the planner to supervise downstream robot learners, addressing their data costs, exploration burden, and lack of long-horizon structure. We develop a taxonomy that provides a common vocabulary for comparing methods across these families and highlights open problems. We argue that *TAMP learning* is a distinct area of study—one that is more than the sum of its parts, just as TAMP itself is more than merely "task planning and motion planning."

Keywords

Task and Motion Planning, Integrated Planning and Learning, Robot Learning

1 Introduction

Autonomous robots should be able to make good decisions quickly with minimal human supervision. They should be generalists, solving a broad range of unforeseen tasks, and specialists, solving familiar tasks with speed and accuracy. In real-world scenarios, robots are often given high-level semantic goals instead of detailed action-by-action instructions. Thus, robots must determine both the sequence of subgoals needed to achieve a task and how to physically execute the corresponding low-level actions in the environment. Task and motion planning (TAMP) addresses these challenges by enabling robots to reason jointly about "what to do" (task planning) and "how to do it" (motion planning). Through this integration, TAMP combines decades of advances in classical AI planning and continuous optimization, generating solutions to problems with long time horizons, constrained geometry, and sparsely specified (e.g., goal-directed) objectives. At the same time, TAMP recognizes that these two levels of reasoning are deeply intertwined: a task plan (e.g., *move to the cabinet, open the door, pick up the package, close the door, move to the hallway*) may or may not be feasible depending on low-level physical details of the environment (e.g., *can the robot fit through the hallway while holding the package?*). This tight coupling has motivated the development of a large family of TAMP techniques (Cambon et al. 2009; Kaelbling and Lozano-Pérez 2011; Srivastava et al. 2014; Toussaint 2015; Dantam et al. 2018; Garrett et al. 2020; Shen et al. 2025).

Despite this progress, traditional TAMP methods remain limited in two fundamental ways. First, *engineering overhead*: TAMP integrates multiple types of domain-specific models, often requiring substantial manual engineering and expertise in both the application domain and TAMP itself. Second, *planning speed*: TAMP can be slow, especially when there are many possible actions, many infeasible task plans, and long task horizons.

In recent years, a substantial body of work has aimed to address these limitations by combining TAMP with machine learning. We organize our discussion around three families of approaches. The first two leverage learning to mitigate TAMP's bottlenecks. *Learning to make TAMP possible* reduces engineering overhead by

[1]Princeton University, [2]Brown University, [3]Allen Institute for Artificial Intelligence, [4]Arizona State University, [*]Equal Contribution

Corresponding author:
Yixuan Huang, Princeton University, Princeton, NJ 08544
Email: yixuan.huang@princeton.edu

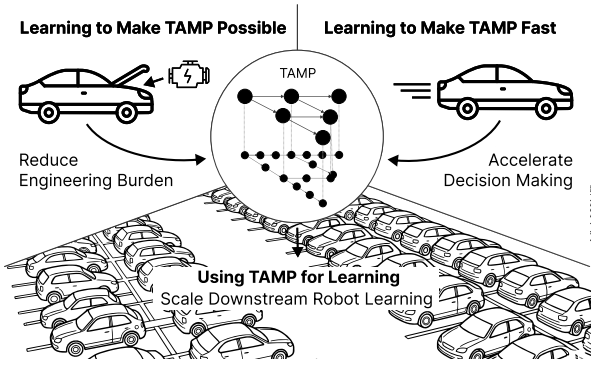

Figure 1. Overview of the survey. We first provide a brief overview of Task and Motion Planning (TAMP). We then review learning to make TAMP possible, learning to make TAMP fast, and using TAMP for learning.

automatically acquiring domain-specific models from data. *Learning to make TAMP fast* accelerates planning by training models from past planning experience. The third inverts the direction. Rather than improving TAMP itself, *Using TAMP for learning* repurposes the planner to supervise downstream learning, mitigating data collection costs, exploration burden, or lack of long-horizon structure, which otherwise limit learning-based approaches. See Figure 1 for an overview.

Our goal is to present a unified survey of these efforts that organizes related work, clarifies the state of the art, and highlights opportunities for future research. There have been multiple recent reviews of TAMP with some discussion of learning as an ancillary consideration (Garrett et al. 2021; Guo et al. 2023; Zhao et al. 2025). In particular, Zhao et al. (2025) discusses aspects of learning in TAMP, and its primary focus is on optimization-based methods. In contrast, our survey emphasizes the bidirectional relationship between TAMP (including both sampling- and optimization-based approaches) and learning. To the best of our knowledge, this is the first to survey the combination of TAMP and learning. Such a survey is overdue, not only to acknowledge the abundance of work, but also to establish *TAMP learning* as a distinct area of study—one that is more than the sum of its parts, just as TAMP itself is more than merely "task planning and motion planning."

As with any survey, we must inevitably draw boundaries in places where the distinctions are fuzzy. In this survey, especially, drawing those boundaries too broadly risks pulling in an unmanageably large and diffuse body of work. For example, TAMP learning is related to (hierarchical, model-based) reinforcement learning (Sutton et al. 1999; Barto and Mahadevan 2003; Klissarov et al. 2025; Moerland et al. 2023); to planning with foundation models (Huang et al. 2022; Zhao et al. 2023; Huang et al. 2023); and to Koopman Operators in control theory (Abraham et al. 2017; Han et al. 2023; Li et al. 2025a). While these areas are outside the primary scope of this survey, we emphasize these connections selectively where they help contextualize developments in TAMP learning. At the same time, we wish to be permissive in our scope and not overly pedantic when defining what qualifies as TAMP learning. As a litmus test for inclusion in this survey, we ask: (1) Is there explicit decision making at multiple levels of abstraction? (2) Does that decision making involve reasoning about the future? (3) Are techniques from either task planning or motion planning incorporated? (4) Is there a robot, broadly construed, simulated or real? Applying that filter leaves a rich but reasonably contained body of work with much to discuss.

In an effort to reach a broad readership, especially those familiar with robotics and machine learning but not necessarily with planning, we begin by introducing TAMP from first principles (Section 2). We then present a taxonomy that organizes TAMP learning along several conceptual axes, providing a common vocabulary for comparing methods throughout the rest of the survey (Section 3). With this vocabulary in hand, we organize our discussion around the three families of approaches outlined above: learning to make TAMP possible (Section 4), learning to make TAMP fast (Section 5), and using TAMP for learning (Section 6). Finally, we discuss open problems in TAMP learning that emerge from the limitations and gaps identified throughout our analysis (Section 7).

2 Task and Motion Planning

We begin with a brief introduction to TAMP and refer readers to other references for details (Garrett et al. 2021; Guo et al. 2023; Zhao et al. 2025). There is no single dominant formalism in the TAMP literature. One tradition, exemplified by PDDLStream (Garrett et al. 2020), formulates TAMP as a single hybrid planning problem in which continuous parameters (e.g., poses, grasps, trajectories) are first-class objects within a symbolic state, and conditional generators called *streams* lazily produce their values during search. A second tradition, which includes bilevel planning (Srivastava et al. 2014) and logic-geometric programming (Toussaint 2015), decomposes the problem into an *abstract* (typically symbolic) layer and a *concrete* (typically geometric and kinematic) layer, with planning proceeding jointly across the two. We adopt the second perspective here because it cleanly exposes components that can be targets for learning, which is our focus. The symbolic layer is likewise not tied to any one language: while we use PDDL-style operators throughout, other approaches express the abstract layer in alternative formalisms, notably temporal logics such as linear temporal logic (LTL) (Kress-Gazit et al. 2009; Plaku and Karaman 2016). However, this section is meant only to establish common vocabulary and intuitions; the body of the survey covers learning methods that operate within other formalisms as well.

States and Actions Consider a decision-making *domain* with states $x \in \mathcal{X}$, actions $u \in \mathcal{U}$, and a

deterministic transition function $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$. A *task* is an initial state $x_0 \in \mathcal{X}$ and a set of goal states $\mathcal{X}_g \subseteq \mathcal{X}$. A plan $\boldsymbol{u} = [u_1, \ldots, u_T]$ is a *solution* if $x_t = f(x_{t-1}, u_t)$ for $1 \leq t \leq T$ and $x_T \in \mathcal{X}_g$. A *planner* produces a plan (or reports failure) given a domain $\langle \mathcal{X}, \mathcal{U}, f \rangle$ and task $\langle x_0, \mathcal{X}_g \rangle$. We are typically interested in a planner's *efficiency* (e.g., planning time and plan length) and *effectiveness* (e.g., success rate) over a distribution of domains and tasks.

This problem formulation makes several simplifying assumptions: fully observable states, deterministic transitions, and goal-based tasks. Many works in TAMP weaken these assumptions; we present the simplified formulation for clarity. Even with these assumptions, planning remains very challenging. In robotics, state and action spaces are typically continuous and high-dimensional; transitions are not always smooth (e.g., due to contact changes); and time horizons are long.

To address these challenges, we introduce an *abstract domain* to accompany the *concrete domain*. Let $s \in \mathcal{S}$ be an abstract state, $a \in \mathcal{A}$ be an abstract action, $F : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ be an abstract transition function, $\alpha : \mathcal{X} \to \mathcal{S}$ be a *state abstractor*, and $\gamma : \mathcal{S} \to \mathcal{P}(\mathcal{X})$ be a *state concretizer*. Given a task $\langle x_0, \mathcal{X}_g \rangle$, we can use the state abstractor to define an abstract task $\langle s_0, \mathcal{S}_g \rangle$ where $s_0 = \alpha(x_0)$ and $\mathcal{S}_g = \{\alpha(x) : x \in \mathcal{X}_g\}$. See Figure 2 for an overview of this coupled concrete-abstract planning system. The key idea is that planning jointly in the abstract and concrete domains can be more efficient than planning in the concrete domain alone. For example, abstract states can be viewed as *subgoals* for concrete planning. However, abstractions are not always helpful: their utility depends on the relationship between the abstract and concrete domains, the task distribution, and the planner itself.

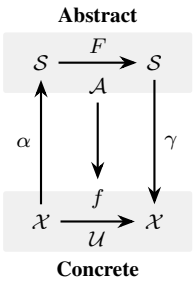

Figure 2. Notation.

Notation 2.1. States and Actions

$x \in \mathcal{X}$	Concrete State
$s \in \mathcal{S}$	Abstract State
$u \in \mathcal{U}$	Concrete action
$a \in \mathcal{A}$	Abstract action
$f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$	Concrete transitions
$F : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$	Abstract transitions
$\alpha : \mathcal{X} \to \mathcal{S}$	State abstractor
$\gamma : \mathcal{S} \to \mathcal{P}(\mathcal{X})$	State concretizer
$x_0 \in \mathcal{X}$	Initial concrete state
$s_0 \in \mathcal{S}$	Initial abstract state
$\mathcal{X}_g \subseteq \mathcal{X}$	Concrete goal states
$\mathcal{S}_g \subseteq \mathcal{S}$	Abstract goal states

Given these abstractions, we can already define a simple hierarchical planning algorithm:

Algorithm 2.1. Naive Bilevel Refinement

Run an uninformed search with initial abstract state s_0, abstract transition function $F : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and abstract goal states $\mathcal{S}_g$ to find an abstract plan $a_1, \ldots, a_T$ inducing the abstract states $s_1, \ldots, s_T$ such that $s_T \in \mathcal{S}_g$. For each abstract transition $s_{t-1} \xrightarrow{a_t} s_t$, search for concrete actions $u_{t,1}, \ldots, u_{t,k_t}$ and states $x_{t,1}, \ldots, x_{t,k_t}$ such that $x_{t,i} = f(x_{t,i-1}, u_{t,i})$ for $1 \leq i \leq k_t$, where $x_{t,0} = x_{t-1,k_{t-1}}$ and $x_{t,k_t} \in \gamma(s_t)$. Backtrack as needed.

Towards Task Planning: Objects and Relations One limitation of the naive strategy above is that the abstract search is uninformed. In task planning, this is addressed by representing states with *relations*. Such relational representations are more compact, generalize across problem instances, and provide useful structure for planning and learning. An object $o \in \mathcal{O}$ has a name (e.g., `plate`) and a type $\lambda \in \Lambda$ (e.g., `dishware`). A *predicate* ψ is a relation that has a name (e.g., `on`) and a type signature (e.g., $\langle$`dishware, surface`$\rangle$). A *ground atom* $\underline{\psi}$ is a predicate and a tuple of objects with types that match the signature (e.g., `on(plate, table)`). An abstract state $s \in \mathcal{S}$ is a set of ground atoms. A ground atom $\underline{\psi}$ is true in state $s \in \mathcal{S}$ if $\underline{\psi} \in s$ and false otherwise.

Notation 2.2. Objects and Relations

$o \in \mathcal{O}$	Object
$\lambda \in \Lambda$	Object type
$D : \Lambda \to \mathbb{N}$	Type-dimension map
$\psi \in \Psi$	Predicate
$\underline{\psi} \in \underline{\Psi}$	Ground atom
C_ψ	Predicate classifier
$\omega \in \Omega$	Operator

There are two common implementations of the state abstractor $\alpha : \mathcal{X} \to \mathcal{S}$. The first performs a direct mapping. For example, a large language or vision-language model can be prompted to return the full abstract state. The second implementation assumes that concrete states $\mathcal{X}$ are also object-centric: each object $o \in \mathcal{O}$ has a state $x(o) \in \mathbb{R}^{D(\text{type}(o))}$, where $D : \Lambda \to \mathbb{N}$ is a type-dimension map specifying the feature-vector dimensionality of each object type. Each predicate ψ is then associated with a relational classifier C_ψ. The state abstractor evaluates all classifiers on all valid object combinations (i.e., types match predicate signature) and returns the set of ground atoms that classify true.

Just as predicates impose relational structure on abstract states, *operators* $\omega \in \Omega$ impose relational structure on abstract actions $a \in \mathcal{A}$ and the transition function $F : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. In the Planning Domain Description Language (PDDL) (McDermott et al. 1998), each operator is given a type signature, a set of preconditions, and effects decomposed into atoms to

add and atoms to delete. For example, a `pick` operator in our dishware domain could be expressed as:

```
(:action pick
  :parameters (?d - dishware ?s - surface)
  :precondition (and (on ?d ?s) (handsfree))
  :effect (and (not (on ?d ?s))
               (holding ?d)
               (not (handsfree))))
```

where `handsfree` and `holding` are additional predicates. Applying an operator in an abstract state whose ground atoms satisfy its preconditions yields a new abstract state by adding the positive effects and removing the negated ones. Decades of classical planning research have investigated how such structure can be exploited to inform search (Bonet and Geffner 2001; Ghallab et al. 2025), for example, by inducing relaxations of the planning problem from which heuristics can be computed (Hoffmann and Nebel 2001). Later, we will see that this same factored structure also provides leverage for learning.

Given this relational structure, we can improve the naive TAMP algorithm above:

Algorithm 2.2. Relational Abstract Planning

Use a task planner (e.g., Fast Downward (Helmert 2006)) with the initial abstract state s_0, operators Ω, and abstract goal states $\mathcal{S}_g$ to generate a ranked set of abstract plans $\left\{[a_1^{(k)}, \ldots, a_{T_k}^{(k)}]\right\}_{k=1}^{K}$. For each abstract plan in order of rank, attempt to refine each abstract transition $s_{t-1} \xrightarrow{a_t} s_t$ into concrete actions and states as in the naive algorithm above. If refinement succeeds for all transitions, return the resulting concrete plan. Otherwise, proceed to the next abstract plan. If all K plans have been exhausted, report failure or request additional abstract plans from the task planner.

Towards Motion Planning: Kinematics and Skills Even if abstract planning is highly efficient, the search over concrete states and actions remains a bottleneck. To make this search tractable, we can impose further structure on the concrete domain. One common set of assumptions gives rise to *multi-modal motion planning* (Siméon et al. 2004; Hauser and Latombe 2010). In the framework above, each abstract state $s \in \mathcal{S}$ already partitions the concrete state space into a region $\gamma(s) = \{x \in \mathcal{X} : \alpha(x) = s\}$—sometimes called a *mode*. To enable motion planning within a mode, we assume that concrete states decompose into a robot configuration $q \in \mathcal{Q}$ and a tuple of object poses $p \in \mathcal{P}$, and that concrete actions are trajectories in $\mathcal{Q}$. Each mode fixes a *kinematic tree* specifying how objects are attached to one another and to the robot—for example, a grasped object moves with the end-effector. Within a mode, the *motion planning problem* reduces to finding a collision-free trajectory in the robot's configuration space between two configurations that satisfy the mode constraints. Motion planners (e.g., PRM (Kavraki et al. 1996), RRT (LaValle and James J. Kuffner, Jr. 2001)) can solve this problem. Transitions between modes (Cambon et al. 2009) correspond to discrete changes in the kinematic tree (e.g., releasing an object). However, not every abstract state transition induces a mode transition; many abstract actions correspond solely to motion within a fixed mode. The abstract plan determines the sequence of modes; concrete planning finds trajectories within each mode and feasible transition points between them.

Notation 2.3. Kinematics and Skills

$q \in \mathcal{Q}$	Robot configuration
$p \in \mathcal{P}$	Object pose
$\sigma \in \Sigma$	Skill
$\mathcal{I}_\sigma$	Initiation set
$\pi(u \mid x)$	Skill policy
Θ_σ	Skill parameters
β_σ	Termination condition
$\delta(\theta \mid x)$	Parameter sampler

A complementary approach introduces *parameterized skills* (Da Silva et al. 2012), which build upon the options framework from hierarchical reinforcement learning (Sutton et al. 1999). A skill $\sigma = \langle \mathcal{I}_\sigma, \pi_\sigma, \Theta_\sigma, \delta_\sigma, \beta_\sigma \rangle$ consists of an initiation set $\mathcal{I}_\sigma \subseteq \mathcal{X}$, a low-level controller π_σ parameterized by $\theta \in \Theta_\sigma$, a parameter sampler δ_σ, and a stochastic termination condition $\beta_\sigma : \mathcal{X} \to \Delta(\{0, 1\})$ (or, when deterministic, a termination set $\beta_\sigma \subseteq \mathcal{X}$). In the TAMP setting, each skill is associated with an abstract action $a \in \mathcal{A}$: the initiation set $\mathcal{I}_\sigma$ is the set of concrete states whose abstract state satisfies the preconditions of a, and the termination condition β_σ corresponds to concrete states whose abstract states reflect the expected effects of a. Throughout this discussion, we assume a has deterministic effects. The skill parameters Θ_σ specify the remaining degrees of freedom—for example, a grasp pose for a pick skill or a placement pose for a place skill. The *sampler* $\delta_\sigma : \mathcal{X} \to \Delta(\Theta_\sigma)$ is a distribution over parameters conditioned on the current state x, from which candidates are drawn and checked for feasibility. Rather than searching directly in the high-dimensional concrete state and action spaces, hierarchical planning reduces to first planning over the set of available skills, then searching over the (typically low-dimensional) parameter spaces of those skills.

We now have enough structure to specialize the general problem setting introduced at the start of this section. A *domain* $\langle \Lambda, D, \mathcal{U}, f, \Psi, \Omega, \Sigma \rangle$ combines an object-centric concrete domain $\langle \Lambda, D, \mathcal{U}, f \rangle$, a relational abstract domain $\langle \Psi, \Omega \rangle$, and a skill library Σ. A *task* $\langle \mathcal{O}, x_0, \underline{\Psi}_g \rangle$ specifies a set of objects $o \in \mathcal{O}$, an initial concrete state $x_0 \in \mathcal{X}$, and goal conditions $\underline{\Psi}_g \subseteq \underline{\Psi}$. Rather than a flat sequence of actions, a solution is now a *bilevel plan* $\langle \boldsymbol{a}, \boldsymbol{\theta} \rangle$ pairing an abstract plan $\boldsymbol{a} = [a_1, \ldots, a_N]$ with parameters $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_T]$, where

each θ_t parameterizes the skill σ_t associated with a_t, such that executing the parameterized skills from x_0 reaches a state x with $\underline{\Psi}_g \subseteq \alpha(x)$.

Given this skill structure, we can refine the concrete planning step of the algorithms above:

Algorithm 2.3. Sampler-Based Skill Chaining

Consider an abstract plan $\boldsymbol{a} = [a_1, \ldots, a_N]$ proposed within the above algorithm. For each corresponding skill σ_t, sample candidate parameters $\theta_t \sim \delta_{\sigma_t}(\theta \mid x_{t-1})$, and check feasibility. If feasible, execute the skill policy $\pi_{\sigma_t}(\cdot \mid \theta_t)$ from x_{t-1} to reach a concrete state x_t satisfying $\alpha(x_t) = s_t$. If no feasible parameters are found after a budget of samples, report the failure to the task planner and replan at the abstract level. Iterate until a fully refined concrete plan is found or a budget is exceeded.

Bridging Between Abstract and Concrete The algorithms above treat abstract and concrete planning as largely independent phases, but in practice, the interface between them is one of the primary technical challenges in TAMP. For example, an abstract plan is not necessarily *downward refinable* (Bacchus and Yang 1994; Marthi et al. 2007): there may be no feasible concrete plan that follows it. Furthermore, the concrete parameters chosen for one abstract action may constrain the feasibility of later actions, e.g., a selected grasp pose may preclude a stable placement downstream.

This boundary between abstract and concrete is also a major focal point in TAMP learning. When learning to make TAMP possible (Section 4), one must learn models that *abstract* from concrete to abstract and *concretize* from abstract to concrete. When learning to make TAMP fast (Section 5), one must learn where and when to allocate planning effort across the two layers. And when using TAMP to accelerate learning (Section 6), one must translate abstract-level guidance into concrete-level learning signals such as demonstrations or reward functions.

3 A Taxonomy for TAMP Learning

The three categories introduced above—learning to make TAMP possible, learning to make TAMP fast, and using TAMP for learning—form the spine of this survey. Several other axes, however, cut across all three categories and serve to unify the field. We focus on axes specific to TAMP learning; considerations common to all machine learning, such as whether learning is supervised or unsupervised, are also important, but they are not unique to TAMP and so we do not include them. See Table 1 for an overview. For each axis below, we identify a principal split and highlight the qualitative and quantitative trade-offs that the split entails.

Data Source TAMP learning, like all machine learning, requires data. We distinguish external data, which originates outside the planner (e.g., demonstration trajectories, oracle predicate classifications, or random skill executions), from planner-generated data, which the planner produces itself, such as motion plans, task plans, or planned skill executions. *Collection cost*: External data is limited by collection costs such as human input, while planner-generated data is limited only by available compute. *Annotations*: Planner-generated data comes with planner-internal annotations essentially for free—abstract plans, sampled continuous parameters, intermediate symbolic states—while external data may carry other useful annotations, such as natural language.

Data Selection Rather than passively ingesting data, TAMP can be used to reason about data selection to improve learning. We distinguish given data, where some process external to the planner chooses what to collect, from planner-selected data, where the planner itself reasons about what data to observe. Data selection is orthogonal to data source: for example, external data can be planner-selected, as in active learning, where the planner chooses which states a human should label, and planner-generated data can be given, as when the planner produces trajectories for an externally chosen set of tasks. *Automation*: Planner-selected data can be collected without human involvement, partially automating the data collection process, whereas external selection often requires human effort. *Coverage*: Planner-selected data concentrates on states the planner already reaches, making each example more informative but leaving gaps where the planner does not yet operate, while given data is independent of the planner's current competence and can be curated to fill such gaps.

Learning Realm Data for learning can be gathered by acting in the physical world or produced entirely in the "mind" of the robot. We distinguish real data, collected by executing actions on physical hardware, from imagined data, produced without physical execution using the planner's models (e.g., a physics simulator, kinematic plans from geometric models, or abstract plans from symbolic models). This distinction is orthogonal to data source: both external and planner-generated data can be real or imagined—demonstrations may be teleoperated on hardware or in simulation, and a planner may collect outcomes on a physical robot or compute them in simulation. *Scalability*: Imagined data can be generated cheaply and in parallel without hardware, whereas real data is limited by the throughput of physical execution. *Fidelity*: Real data reflects the true dynamics of the environment, whereas imagined data is limited by the simplifying assumptions of the planner's models.

Learning Process A distinctive feature of TAMP learning is that both the task planner and motion planner are available during training, not only at deployment. We distinguish decoupled learning, in which data is

Dimension	Principal split	Qualitative & Quantitative Trade-offs
Data Source	External ↔ Planner-generated	Collection cost · Annotations
Data Selection	Given ↔ Planner-selected	Automation · Coverage
Learning Realm	Real ↔ Imagined	Scalability · Fidelity
Learning Process	Decoupled ↔ Interleaved	Compute · Adaptivity · Stability
Learned Representations	Native ↔ Latent	Interpretability · Reuse · Expressiveness
Formal Guarantees	Preserving ↔ Abandoning	Failure modes · Stakes

Table 1. A taxonomy of TAMP learning. Each row identifies a dimension along which methods differ, the two principal options for that dimension, and the trade-offs that make the choice consequential.

collected once, and the learner is trained without further reference to the planner, from interleaved learning, in which the planner is invoked repeatedly over the course of learning. This dichotomy is related to but distinct from the standard online/offline distinction in machine learning: interleaved learning is a form of online learning in which the loop runs through a planner, whereas decoupled learning encompasses both offline learning and online learning that does not invoke the planner. *Compute*: Interleaved learning pays the planner's cost throughout learning, while decoupled learning pays it only during data collection, if at all. *Adaptivity*: Interleaved learning can adaptively target the learner's weaknesses, improving the value of each training example, while decoupled learning is committed to whatever was collected up front. *Stability*: Interleaved learning with adaptive selection must contend with a non-stationary target—the learner's behavior shifts the distribution of training data—and is often harder to optimize than its decoupled counterpart.

Learned Representations TAMP planners were originally designed to operate on representations that humans can read and write. TAMP learning therefore faces a choice: stay with these native representations and learn to ground them to raw observations and to refine the planner's outputs into low-level actions, or move to latent representations (e.g., learned continuous embeddings whose semantics are not explicitly specified), rebuilding the planner around them. *Interpretability*: Native representations are human-readable, making intermediate states, plans, and failures easy to inspect; latent representations are opaque by default. *Reuse of classical machinery*: Native representations inherit decades of planning algorithms; latent representations must reinvent or approximate them. *Expressiveness ceiling*: Native representations are limited by structural biases and what their designers anticipated; latent representations can, in principle, capture phenomena the designer did not foresee.

Formal Guarantees Many classical TAMP planners offer formal guarantees: soundness, probabilistic completeness, or asymptotic optimality. Incorporating learning forces a choice about whether to maintain these properties. We distinguish preserving methods, in which the learner is structured so that these guarantees still hold, from abandoning methods, in which guarantees are sacrificed for the sake of other considerations, e.g., planning efficiency. *Failure modes*: Preserving methods fail gracefully when the learner is wrong, while abandoning methods can fail silently or catastrophically. *Stakes*: Preserving methods can be deployed in settings where formal guarantees are required (e.g., safety-critical, regulated, or certified domains); abandoning methods typically cannot.

4 Learning to Make TAMP Possible

A major bottleneck in using TAMP is that many models must be specified, typically by a human engineer who is an expert in both the application domain and in TAMP itself. In this section, we discuss *learning to make TAMP possible*, where these components are learned from data. We organize the section by the major components that may be learned (Figure 3): the state abstractor α, the skill library Σ, the abstract transition function F, and the concrete transition function f. Within each subsection, we ask *where does the corresponding component come from?*, identifying positions in the literature whose answers determine what data is required, what supervisory signal is used, and how the learned component relates to the rest of the TAMP system. Not every approach uses every component, and several works learn more than one; we discuss each in the subsection that captures its main contribution.

4.1 Learning Predicates

Abstraction learning is a challenging and long-standing topic (Li et al. 2006). The challenge arises because abstraction is the point at which raw, continuous, unstructured observations of the world must be converted into compact, structured, and often discrete representations (van den Oord et al. 2017; Hafner et al. 2021); in other words, learning an abstractor amounts to learning to cross the symbolic/sub-symbolic boundary (Smolensky 1988). What makes the problem distinctive

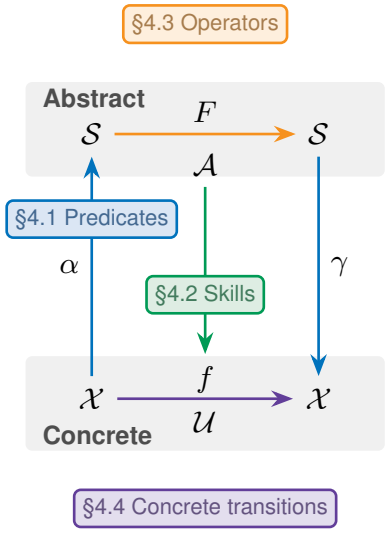

Figure 3. The components learned in Section 4, located within the notation of Figure 2. Each subsection asks where one component comes from: the state abstractor α (predicates), the skill library that realizes abstract actions concretely, the abstract transition function F (operators), and the concrete transition function f.

in TAMP learning is that the abstractions must be usable by a TAMP planner and, when used by one, must lead to efficient and effective planning.

In this section, we focus on state abstractions that take the form of predicates—factored and relational structures over typed objects, introduced in Section 2. Specifying α then reduces to specifying a vocabulary of predicates Ψ and their classifiers $\{C_\psi\}_{\psi\in\Psi}$, with $\alpha(x) = \{\underline{\psi} \in \underline{\Psi} : C_{\underline{\psi}}(x) = 1\}$. This factored, relational structure serves two purposes: it allows the planner to exploit the abstraction directly, and it supports learning by enabling generalization across problems with different objects. Focusing on predicates also amounts to a commitment to *native* representations in the sense of Section 3: predicates are discrete, composable, and inspectable. A separate body of work, in hierarchical and model-based reinforcement learning, learns abstract states that are fully latent (Hafner et al. 2021; Schrittwieser et al. 2020), sacrificing interpretability and compatibility with classical planners for greater expressiveness.

We identify four positions in the literature on where predicates originate.

Predicates as latent state-space structure. Predicates are ultimately state abstractions, and one answer to where they come from is that they should be derived directly from states. Shah et al. (2025) consider a motion-planning-specific instantiation: states are assumed to be defined by objects and their poses, and geometry is the salient structure to be abstracted. The method does not require any predefined predicates and learns both predicates and their associated classifiers directly from geometric observations. The method clusters relative object poses across expert demonstrations and converts densely occupied regions of the relative-pose space into predicates whose classifiers test geometric configuration. For example, configurations in which one object rests stably on another form a dense cluster in relative-pose space, yielding a predicate such as `on(o1, o2)` whose classifier tests whether the observed pose lies in that cluster. Asai et al. (2022) take a different approach that does not assume object-centric states: predicates are learned directly from images via a state autoencoder trained for reconstruction. A predicate is good to the extent that it carries information needed to reconstruct the lower-level observations. Dittadi et al. (2021) take a similar autoencoder-based approach. In these cases, the stance is that predicates should capture the salient information apparent in states alone.

Predicates as the effects of action. Another answer to where predicates come from is that they should be derived from the applicability and effects of skills on states. Konidaris et al. (2018) consider a theoretical instantiation: given a fixed set of skills, there exists a coarsest abstraction of the state space that suffices for evaluating any plan composed of those skills, and the symbols correspond to equivalence classes of states that the skills cannot tell apart. The construction is provably sound and complete for planning with the given skills. For example, the set of states from which a `pick` skill can be initiated, and the set of states it terminates in, each becomes a symbol that the planner must track. This tradition has been extended in several directions (Andersen and Konidaris 2017; Ames et al. 2018; Xu et al. 2021; James et al. 2022; Pacheck et al. 2023). Ahmetoglu et al. (2022) take a different approach that learns these distinctions from interaction: a robot interacts with objects using a fixed action repertoire, and an encoder–decoder with a binary bottleneck is trained to predict the effects of each action on the next observation. Categories such as `rollable` or `insertable` emerge because they are the distinctions needed to predict what actions can do, building on earlier interaction-driven clustering approaches (Ugur and Piater 2015). Whereas these methods derive symbols from a fixed skill repertoire, Shao et al. (2026) invent predicates and skills jointly from unsegmented, unlabeled play data: demonstrations are split into reaching and manipulation phases, each skill is fit as a stable dynamical-system controller, and predicates are formed by clustering the relative object poses at which manipulation phases terminate, so that each predicate marks a configuration some skill reliably reaches. Yang et al. (2026) extend the same idea to black-box skills mediated by a foundation model: when two executions of the same skill produce different outcomes under the current vocabulary, a vision–language model proposes a new predicate that distinguishes them, again with soundness and probabilistic-completeness guarantees. Across these approaches, the stance is that predicates should capture distinctions induced by skills, because

the predicates' purpose is ultimately to determine whether sequences of skills can be executed.

Predicates as planning instruments. Another answer to where predicates come from is that they should be chosen for their utility to a planner. Silver et al. (2023) consider a search-based instantiation: candidate predicates are drawn from a grammar over object features, and a hill-climbing search retains those that most reduce a surrogate of expected planning time on held-out tasks. For example, a candidate predicate `near(o1, o2)` is kept if it accelerates planning on the task distribution; otherwise it is discarded. This overall optimization-based approach is similar to Jetchev et al. (2013), which is itself related to the earlier rule-learning framework of Pasula et al. (2007). Li et al. (2025b) take a different approach in which predicates are neural classifiers, trained jointly with their symbolic operators in a bilevel optimization. This relaxes the expressiveness limits of a finite grammar and enables the discovery of predicates over high-dimensional inputs such as SE(3) poses and point clouds. In both cases, the stance is that predicates should be judged by planning utility.

Predicates distilled from foundation models. Another answer to where predicates come from is that they should be distilled from a foundation model. Pretrained on human-generated text and images, a foundation model encodes an implicit prior over which categories distinguish states, with the ultimate source of that prior being human concepts about the world. This commits predicates to representations the model can produce—natural language or code—but most approaches augment this commitment with another signal, typically planning utility, that filters the predicates the model proposes. Liang et al. (2025, 2026) consider a programmatic instantiation: a vision–language model is prompted in three ways—discrimination of skill applicability, transition modeling on before–after image pairs, and unconditional extensions of the existing vocabulary—to produce predicates as Python programs that may themselves invoke VLM perception calls. For example, the model might propose a predicate `graspable(o1)` as a Python function that queries the VLM with "is this object graspable in the current image?" Athalye et al. (2026) take a different approach in which the VLM proposes a large pool of candidate predicates, and an optimization-based model-learning procedure selects a compact subset that best explains a set of demonstrations. Han et al. (2024) use human language feedback during interaction to drive an LLM's proposals; the planner validates the resulting predicates by attempting to solve tasks. Byrnes et al. (2025) continue this loop across tasks, accumulating a predicate vocabulary over time. In all cases, the foundation model supplies the prior, and another signal—planning, demonstrations, or language feedback—selects which of its proposals survive.

4.2 Learning Skills

Not all TAMP approaches require learned skills: in many approaches, what bridges the abstract and concrete levels is a motion planning or optimization procedure rather than a skill (Srivastava et al. 2014; Toussaint 2015; Garrett et al. 2020). Other TAMP approaches use skills as primitives, and making those skills available is itself part of making TAMP possible.

In this section, we focus on the skill policy π_σ of parameterized skills.* What makes skill learning distinctive in TAMP is that the planner's symbolic abstractions are typically *lossy* (Chitnis et al. 2016; Silver et al. 2022): many low-level states map to the same abstract state, and which one a skill should produce depends on what comes next in the plan. The continuous parameter space Θ_σ supplies this flexibility, and also controls aspects that the symbolic level does not capture, e.g., execution speed.

We identify four positions in the literature on where skill policies originate.

Skills learned from predicates. One answer is that predicates come first: a predicate vocabulary is learned or supplied externally, and skill policies are trained to realize specified symbolic effects. Methods in this stance fall along the decoupled vs. interleaved *Learning Process* axis of Section 3, depending on whether the planner is invoked during skill learning. Cheng and Xu (2023) take the interleaved side: a symbolic task planner decomposes the task and is repeatedly invoked during training to choose what to refine next, while reinforcement learning trains each skill against a reward derived from the corresponding symbolic effect. For example, a "place" skill is trained against a reward derived from `on(o1, o2)`. Illanes et al. (2020) and Sarathy et al. (2021) take related interleaved approaches: the former uses symbolic plans to shape RL rewards, the latter triggers RL skill discovery when planning fails. Related approaches integrating symbolic planning with RL for skill learning include (Yang et al. 2018; Lyu et al. 2019; Kokel et al. 2021; Mayr et al. 2022; Lorang et al. 2025b; Liu et al. 2024a; Urbaniak et al. 2021). On the decoupled side, Silver et al. (2022) learn parameterized policies from demonstrations alongside operators and samplers; Achterhold et al. (2023) learn skill policies via intrinsic-motivation reinforcement learning given a predefined predicate vocabulary, with the planner invoked only after training. Abbatematteo et al. (2024) learn structured, sustained-contact manipulation skills by using free-space motion planning to compose skills, thereby simplifying the learning problem. Paxton et al.

*The other skill components—the initiation set $\mathcal{I}_\sigma$, parameter sampler δ_σ, and termination condition β_σ—are either tied to operators (covered in Section 4.3) or to planning efficiency. Sampler learning in particular sits between making TAMP *possible* and *fast*: samplers can be hand-coded but learning them primarily benefits planning efficiency, so we discuss sampler learning in Section 5.

(2017) learn both the low-level control policies and the task-level option policies with reinforcement learning against linear temporal logic specifications, composed by a Monte Carlo tree search. In all of these, the predicate vocabulary (i.e., termination conditions) is given, and skill learning fills in the implementations.

Skills come first; predicates emerge. A second answer reverses the order: skill policies come first, and predicate structure (if any) emerges from their effects. Unsupervised skill discovery is itself a large subfield of hierarchical reinforcement learning (Pateria et al. 2021; Klissarov et al. 2025); here we focus on methods that explicitly interface with a TAMP planner. Bagaria and Konidaris (2020) use deep skill chaining to grow a library of options backward from goal states, with each new option initialized in a region where an earlier one can be reliably invoked; this builds on the original skill-chaining framework of Konidaris and Barto (2009). Da Silva et al. (2012) similarly learns parameterized skill policies across a task family via reinforcement learning. From the imitation side, Zhu et al. (2022) discover skills from unsegmented demonstrations by clustering recurring trajectory patterns. This view is the dual of the action-effects stance for predicates (Section 4.1): the same camp that says "predicates come from skills" here says "skills come first."

Skills and predicates co-invented. A third answer rejects the primacy question: predicates and skill policies should be co-invented from the same data. Shao et al. (2026) realize this directly: given unlabeled, unsegmented demonstrations, the method jointly learns a vocabulary of relational predicates, a library of parameterized skill policies, and the symbolic operators that link them. Keller et al. (2025) take a related neuro-symbolic imitation-learning approach in which a symbolic representation decomposes a task and neural skill policies are trained against that decomposition. Lorang et al. (2025a) extends the joint-learning idea to few-shot settings with diffusion-policy controllers, and Liu et al. (2024b) includes natural-language annotations as a third learnable component.

Skills as commands to a generalist controller. A fourth answer takes the skill to be a command rather than a learned policy. The base controller is a generalist vision–language–action model, and a skill is whatever natural-language command the higher-level controller issues to it. The skill library is implicit in the executor's interpretation; the open design question is what to command. Shi et al. (2025) take a hierarchical instantiation in which a high-level model produces commands for a lower-level VLA executor. In Plan-Seq-Learn (Dalal et al. 2024), an LLM produces a plan as a sequence of natural-language steps that reinforcement learning fills in.

4.3 Learning Operators

Not all abstract world models are operator-based. Huang et al. (2024a) jointly learn relational classifiers and a latent-space transition model from partial-view point clouds, treating the dynamics as a learned latent function conditioned on both discrete and continuous parameters of the skills rather than symbolic operators; related approaches include JEPA-style world models (Assran et al. 2025) and program-synthesized transition models (Ahmed et al. 2025). In this section we focus on operator-based abstract world models, since these are what classical TAMP planners consume directly.

Symbolic operator learning has a long history in classical planning (Arora et al. 2018). Operator learning for TAMP differs in two ways. First, operators must connect to continuous low-level dynamics, and are accordingly *lossy*; TAMP-aware operator learning therefore typically adopts *weak semantics* (Marthi et al. 2007; Chitnis et al. 2022): operators describe what *some* matching low-level state transitions to, rather than what *all* matching states transition to, with the gap filled by bilevel planning with backtracking. Second, operators need not capture every effect of a skill, only those required for planning (Kumar et al. 2023): a `pick` skill is intended to make `Holding` true, while incidental changes like `HandEmpty` becoming false are modeled only when chaining requires them.

We identify four positions in the literature on where operators originate.

Operators learned given predicates. A predicate vocabulary is given; operators are inferred from observed transitions in the abstract space. Silver et al. (2021b) provide a direct formulation: clustering lifted effects across transition data and learning preconditions per effect set. Chitnis et al. (2022) extend this to a fully neuro-symbolic transition model that learns symbolic operators alongside neural samplers and a low-level transition function. Silver et al. (2022) learn operators alongside policies and samplers from demonstrations. Huang et al. (2025a) infer a task-specific domain at test time from a few demonstrations, using a learned estimator to propose candidate operators and a search procedure that prunes them to a compact, useful set. Earlier work in this tradition includes the probabilistic relational rule-learning of Pasula et al. (2007), the symbolic-effect learning over predefined predicates in Achterhold et al. (2023), and the operator generation from segmented demonstrations in Diehl et al. (2021). A separate question is how to collect informative transition data when none is given: Chitnis et al. (2021b) address this with goal-literal babbling, an interleaved exploration scheme that samples relational goals and uses the current operator model to plan toward them, with provable convergence to the true model.

Operators derived from skills. Skills are given; operator schemas fall out of skill structure. Konidaris

et al. (2018) derive operators analytically from skill initiation sets, effect sets, and state masks, yielding a propositional PDDL-like representation with provable grounding properties. This stance is notable for *preserving* the classical-planning guarantees of Section 3, rather than abandoning them. The constructivist line continues with Andersen and Konidaris (2017), who add active exploration when no initial dataset is given; Ames et al. (2018), who extend the construction to parameterized motor skills; James et al. (2022), who learn portable, object-centric symbolic vocabularies that transfer across tasks; and Xu et al. (2021), who use the learned operator structure to predict TAMP plan feasibility from observations. A related line treats the agent's skills as a black-box interface and derives operators by querying behavior: Verma et al. (2022) estimates an interpretable operator model by issuing targeted queries to a black-box planning agent.

Operators given first; predicate groundings learned. The predicate vocabulary is declared as part of the operator schemas, but the interpretation of each predicate on raw observations is learned downstream. Liu et al. (2024b) use a large language model to generate operator schemas from language-annotated demonstrations, then train neural classifiers that ground the named predicates from perception. Migimatsu and Bohg (2022) take operator pre- and post-conditions as predefined and learn visual classifiers for the constituent predicates indirectly from action examples. Mao et al. (2022) introduce a domain-definition language in which users write operator sketches over predicates, leaving neural networks to fill in the predicate evaluations.

Operators co-invented with the rest. The joint-learning methods covered in Sections 4.1 and 4.2 also yield operators as outputs: Shao et al. (2026), Li et al. (2025b), Keller et al. (2025), Yang et al. (2026), Shah et al. (2025), and Byrnes et al. (2025) all learn operators alongside predicates and/or skills, by inferring effect changes from data or by generating operator schemas with a foundation model. Ahmetoglu et al. (2022) takes a related approach with a distinctive representational choice: a neural binary-bottleneck transition model is distilled into PPDDL operators.

4.4 Learning Concrete Transition Models

Not all TAMP methods need a low-level world model. PDDLStream (Garrett et al. 2020), for example, replaces forward simulation of f with conditional generators (streams) that produce continuous parameter values satisfying geometric feasibility constraints; the planner reasons over certified facts rather than predicted transitions. For TAMP approaches that do require f, we identify three positions on where the model comes from; only some involve learning.

From physics simulation. TAMP systems that need a forward dynamics model traditionally use a hand-coded physics simulator such as PyBullet or MuJoCo, sometimes made differentiable to enable joint optimization over discrete plan structure and continuous parameters (Toussaint et al. 2018; Shen et al. 2025; Lee et al. 2025b). In these settings the dynamics are analytic, not learned; what is at stake is aligning the symbolic abstractions with the simulator. A learning variant could in principle replace the analytic simulator: Allen et al. (2023) show that graph network simulators can capture the discontinuous rigid-contact dynamics characteristic of contact-rich manipulation. Whether such learned physics simulators can support full TAMP-style planning remains an open question.

From large-scale world models. A more recent direction is to use a pretrained or large-scale learned world model in place of an analytic simulator. Assran et al. (2025) train a video world model on web-scale data and use it for zero-shot robotic planning with visual subgoals. Zhang et al. (2026) extend this line with hierarchical planning over latent world models at multiple temporal scales (Chen et al. 2026). Huang et al. (2026) scale 3D particle-based world models for in-the-wild manipulation, and Wang et al. (2026) build an interactive video simulator for training policies.

Learned domain-specifically with the rest of TAMP. Other low-level world models are learned alongside the predicates, skills, and operators, from the same limited experience that drives the rest of TAMP. Chitnis et al. (2022) pair each operator with a neural low-level transition function over object-centric attributes. Agia et al. (2023) learn per-skill dynamics and Q-functions for planning-time feasibility prediction. Xia et al. (2019) learn sparse relational transition models that select a small set of relevant objects per action. Ahmed et al. (2025) take a distinctive route by synthesizing low-level transitions as Python programs with an LLM, refining the code when predictions diverge from observations.

A common theme across these four subsections is the lack of consensus about where each component—predicates, skills, operators, the concrete transition function—should come from, and in what order. Each subsection identified multiple positions on its own primacy question, with the same camps sometimes reappearing across subsections from different angles. The field has not converged on a single recipe for what to learn first and what to assume given—nor does it necessarily need to. In some settings it is easier to hand-engineer predicates and learn skills; in others, hand-coded skills and learned predicates are the more practical choice; the right ordering often depends on what is readily available for hand-engineering in the application at hand.

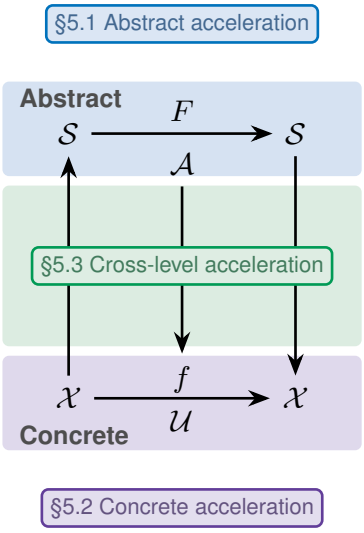

Figure 4. The acceleration targets of Section 5, organized by where the learned model operates relative to the abstract/concrete boundary of Figure 2: the abstract side, the concrete side, or the boundary crossing the two.

5 Learning to Make TAMP Fast

Even when all the models required for TAMP are in place, planning itself can be prohibitively slow: motion planning is slow in high-degree-of-freedom configuration spaces, task planning is slow when there are many objects and long horizons, and combining the two adds the further cost of repeatedly choosing which task plans to attempt to refine, with refinement failures feeding back into that choice. In this section we discuss *learning to make TAMP fast*, where past planning experience—typically collected on easier problems—is used to train models that accelerate planning on harder ones. We organize the section by where the learned model operates relative to the abstract/concrete boundary (Figure 4): on the abstract side (Section 5.1), on the concrete side (Section 5.2), or crossing the two (Section 5.3). Some methods fall into more than one category; we discuss each in the subsection that captures its main contribution.

5.1 Abstract Acceleration

Three families of methods accelerate the abstract planner. The first learns policies over abstract actions, the second learns to decompose the problem into shorter subproblems, and the third learns to reduce what the planner must consider.

Abstract policies. One way to accelerate symbolic planning is to learn a policy over abstract actions, either to bypass search entirely or to bias it toward promising plans. Driess et al. (2021b) bypass search: a convolutional recurrent network maps an image and a symbolic goal to an abstract action sequence, which a logic-geometric programming solver then refines. Sun and Shoukry (2024) likewise learn neural planners that bypass search, here for linear-temporal-logic tasks, but train them against the LTL automaton so that the learned planner inherits its correctness guarantees. Curtis et al. (2022) bias search instead: their GENTAMP framework learns a generalized policy over feature evaluations and compiles it into domain constraints that bias a TAMP solver. Khodeir et al. (2023b) use a graph-neural-network policy as a priority function during PDDLStream skeleton search; refinement failures update the priorities online, so the method also acts cross-level (Section 5.3). Lee et al. (2025a) use a large language model to supply a prior over abstract actions that warm-starts a tree search over plan skeletons. These methods occupy different positions on the formal-guarantees axis (Section 3): pruning the search space abandons completeness, methods that only re-order it preserve it, and neurosymbolic designs can preserve guarantees by construction even while bypassing search.

Subgoal decomposition. A second family replaces a long-horizon problem with a sequence of shorter subproblems. Because TAMP solving time grows exponentially with plan length, decomposition yields large speedups and lets the planner replan locally on failure. Moreno et al. (2024) specify decompositions by hand using *sketches*, a rule-based classical-planning language that partitions a problem by state features. Other methods propose the decomposition with a large language or vision-language model at inference time: Yang et al. (2025) generate VLM subgoals and solve each with a TAMP planner that inserts additional actions, re-prompting on geometric infeasibility; Kwon et al. (2025) similarly decompose with an LLM, switching between a symbolic planner and an MCTS-based LLM planner by subgoal complexity. A third approach learns the decomposition: Zhang et al. (2025) mine subgoal sequences from TAMP demonstrations and use a graph neural network at runtime to identify the closest mined subgoal. Resolving or eliminating dependencies between subgoals remains a core challenge.

Task reduction. A third family shrinks the planning problem itself, reducing the set of objects or the abstract structure the planner must consider. Silver et al. (2021a) score object relevance with a graph neural network and plan with only the high-scoring objects, falling back to broader sets if planning fails. Zhang et al. (2025) incorporate geometric features into the relevance predictor and combine object reduction with their learned subgoal decomposition. Rana et al. (2023) use an LLM to perform a *semantic search* over a hierarchical 3D scene graph, extracting a task-relevant subgraph that a downstream planner operates on. Chitnis et al. (2021a) take a different angle: their CAMPs framework learns to impose constraints on the agent's behavior so that context-specific independences shrink the per-context abstract MDP. Yan et al. (2025) derive analogous constraints

from vision-language reasoning, ruling out actions whose downward refinement is likely to fail.

5.2 Concrete Acceleration

Three families of methods accelerate the concrete planning step. The first replaces a hand-crafted parameter sampler with a learned generative model. The second leaves a base sampler intact and learns a feasibility model that filters its proposals. The third composes per-skill models over an entire plan skeleton, optimizing all continuous parameters jointly.

Direct generative samplers. One strategy is to learn a generative model that directly produces continuous concrete values—skill parameters, stream values, or bindings, depending on the TAMP formalism. Kim et al. (2018) train a generative adversarial network on samples from prior TAMP search, using importance-ratio estimation to correct for the abundance of off-target samples. Fang et al. (2024) use diffusion models with classifier-based guidance, drawing samples conditioned on goal predicates such as DOORCLOSED or DOOROPEN; objects are represented as latent embeddings of point clouds, enabling generalization across articulated-object instances. Ortiz-Haro et al. (2022) pair a deep generative model with an analytic constraint term and a final projection to the constraint manifold, factorizing across actions for sequential manipulation. Two extensions adapt this family across task distributions: Mendez-Mendez et al. (2023) learn a lifelong mixture of generative parameter models with online selection driven by auxiliary-task confidence, and Chitnis et al. (2019) meta-learn modular sampler "specializers" across diverse manipulation tasks for fast adaptation. Across this family, the parameter sampler is learned as a latent generator—the latent end of the Representations dichotomy (Section 3), trading interpretability for expressiveness over high-dimensional and structured parameter spaces.

Constraint-model samplers. An alternative is to keep a hand-crafted base sampler—an inverse-kinematics solver, an antipodal-grasp sampler—and learn a model of where its proposals are feasible, using that model to filter or reweight them. Wang et al. (2021) train Gaussian-process scoring functions over skill parameters via active level-set estimation and reject samples outside the estimated super-level set; the learned samplers integrate into PDDLStream (Garrett et al. 2020) as drop-in replacements for hand-crafted streams, demonstrated on dynamic manipulation skills (pouring, scooping) whose feasibility conditions are impractical to specify by hand. Ren et al. (2022) build a library of transferable constraint primitives from geometric backtracking during TAMP, then use Bayesian optimization over these primitives to focus the binding planner's search in new tasks. Kumar et al. (2024) learn per-skill energy functions that rank candidates from a hand-crafted parameter prior; in parallel, per-skill *competence* estimates—each skill's expected success rate under its current parameter policy—drive both skeleton selection and the choice of which skill to practice next during free time. Cieślar et al. (2025) extend this strategy to long-horizon dependencies by conditioning a transformer-based feasibility classifier on both the current concrete state and the remaining plan suffix; the classifier rejects samples that cannot lead to successful refinement and prioritizes which earlier parameter choices to revisit during backtracking. Because the base sampler remains intact, these methods inherit its support and preserve whatever completeness guarantees it provides; the learned model adds evaluation cost in exchange for a tighter sampling distribution.

Joint skeleton composition. The methods above learn samplers for individual actions; the planner then composes them sequentially during refinement. Mishra et al. (2023) propose an alternative in which composition is performed within the generative modeling framework itself, training an unconditional diffusion model for each skill that captures the joint distribution over pre-state, action parameters, and post-state. Given a plan skeleton, the per-skill score functions are coupled at shared state variables, allowing all skill parameters to be sampled in parallel via a single reverse diffusion process; geometric constraints are incorporated through classifier-based guidance during denoising. Agia et al. (2023) apply the same compositional idea to per-skill Q-functions: Q-values are multiplied along a candidate skeleton to yield a joint feasibility objective over all skill parameters (the same paper appears in Section 4.4 for its dynamics-based feasibility role). Xue et al. (2024) compose per-skill value functions instead, using tensor-train factorization for tractable representation and a first-order mathematical program that alternates between symbolic skeleton search and continuous skill-parameter optimization to maximize the sum of value functions across the plan. All three methods train per-skill objects independently and assemble them at planning time, sidestepping the cost of training over joint distributions but assuming that the per-skill abstractions chain cleanly across the skeleton—an assumption that breaks when cross-skill dependencies span features not visible during single-skill training.

5.3 Cross-Level Acceleration

Three families of methods couple the two levels. The first predicts the feasibility or cost of candidate actions or plans from continuous evidence. The second learns evaluation functions over the hybrid space of partial plans paired with their geometric state. The third revises the high-level plan in response to motion-refinement failures observed during planning.

Feasibility prediction. A learned model predicts whether a candidate action or plan will refine successfully, given the current concrete state, and the planner uses the prediction to prune or rank candidates before invoking expensive refinement. Early work (Wells et al. 2019) introduced a per-action feasibility classifier; subsequent extensions enriched the input representation and broadened the problem types (Driess et al. 2020; Xu et al. 2022; Bouhsain et al. 2023). Yang et al. (2023) predict full-plan feasibility with a Transformer; Noseworthy et al. (2021) actively select which plans to evaluate during training. Beyond binary feasibility, learned models can score plans continuously: Luong (2023) predict refinement time with a graph neural network, and Dhakal et al. (2026) anticipate the cost of subsequent tasks.

Search heuristics over hybrid representations. Rather than evaluating fully constructed action sequences, a learned model can guide search through the hybrid space of partial plans paired with their geometric state. Chitnis et al. (2016) introduced this approach with a linear value function over plan-refinement-graph features, trained by inverse reinforcement learning from expert planning traces. Subsequent work replaced the linear model with graph neural networks: Kim et al. (2022) learn a heuristic that prioritizes feasibility checks on promising state-action pairs in geometric TAMP; Bradley and Roy (2022) learn when to attempt expensive sub-problems during long-horizon search; and Khodeir et al. (2023a) learn a heuristic for prioritizing object and fact expansion in PDDLStream-based TAMP. Kim et al. (2019) take a complementary route, representing each problem instance in a *score-space* defined by the performance of the solutions tried so far, and transferring learned search-space constraints to instances similar in that space. A more recent direction uses large language models as the high-level planner with learned per-skill scoring as a feasibility filter: Lin et al. (2023)'s Text2Motion combines an LLM-proposed skeleton with STAP-based geometric feasibility (Section 5.2), and Huang et al. (2025c)'s Points2Plans pairs LLM-proposed plans with a learned relational dynamics model for parameter instantiation. A related line forgoes the symbolic planner altogether, prompting a foundation model to emit a constraint problem that a continuous solver optimizes into robot motions: Curtis et al. (2025) generate a continuous constraint satisfaction problem from an LLM and solve it by sampling, while Huang et al. (2025b)'s ReKep has a vision–language model produce relational keypoint constraints solved by optimization.

Failure-driven high-level revision. Rather than predicting feasibility or scoring plans before refinement begins, a third family revises the high-level plan in response to motion-refinement failures observed during planning. Sung et al. (2023) learn backjumping heuristics that identify which earlier decision caused a refinement failure and skip ahead during backtracking; Kwon and Kim (2026) instead let a vision–language model choose where to backtrack, guided by visual renderings of candidate states in a kinodynamic TAMP search. Wang et al. (2024)'s LLM3 prompts an LLM with motion-failure information to revise the abstract plan. Duan et al. (2025) train a vision-language model (AHA) dedicated to failure detection and reasoning; its failure explanations have been used to drive high-level plan revision in TAMP. LAZY (Khodeir et al. 2023b), whose abstract policy is discussed in Section 5.1, updates its policy's action priorities online whenever motion refinement fails.

A common theme across these three subsections is that the families are largely complementary: a TAMP system could in principle use a learned abstract policy to propose skeletons, learned samplers to generate continuous parameters, and a learned feasibility predictor to filter candidates before refinement. Few systems combine more than one or two of these strategies, however, and how to compose them effectively—which combinations exhaust the gains from learning, which are redundant, which interfere with each other—remains largely unexplored.

6 Using TAMP for Learning

Sections 4 and 5 use learning to address TAMP's bottlenecks. This section inverts the direction. The bottlenecks of robot learning—the data costs of demonstrations, the exploration burden of reinforcement learning, and the lack of long-horizon structure that learning struggles to discover from data alone—are addressed by repurposing the planner as a source of supervision for downstream learners. We organize the section by three modes of integration. *Distillation* (Section 6.1) trains a standalone learner to imitate or replace the TAMP pipeline, removing TAMP at runtime. *Scaffolding* (Section 6.2) keeps TAMP at deployment and uses learning to extend its capabilities with new components—skills, recovery policies, learned action primitives. *Active learning* (Section 6.3) uses planning to allocate data collection effort to improve existing components.

6.1 Distillation

TAMP solutions are slow at runtime, depend on precise state estimation, and produce per-task plans that do not generalize. *Distillation* addresses these limitations by training a standalone learner to imitate or replace the TAMP pipeline: TAMP runs at training time, generates supervision, and is absent at deployment. We identify three families, distinguished by what the learner inherits from TAMP.

Hierarchical distillation. One family trains hierarchical policies that retain TAMP's two-level decomposition. Driess et al. (2021a) learn a hierarchical

visuomotor policy from logic-geometric-programming solutions: a high-level network jointly predicts plan feasibility via cost-to-go and parameters for low-level energy-function controllers, all from object-masked images. McDonald and Hadfield-Menell (2022) train a feed-forward task-level policy alongside operator-specific controllers, with task plans supervising the top level and motion plans supervising the bottom. Both methods bypass iterative TAMP refinement at execution time while retaining the task/motion structure in the learned policy.

End-to-end policy distillation. A second family flattens the hierarchy: a single policy maps raw sensor observations directly to actions, trained on planner-generated trajectories via behavior cloning. Dalal et al. (2023) train Transformer policies on large-scale TAMP-generated demonstrations, curating the dataset to reduce its multi-modality through cost-sensitive planning, trajectory smoothing, consistent IK seeding, and outlier filtering. Jung et al. (2025) extend RRT into the space of skills and object goals, distilling Skill-RRT solutions into a flat policy that handles long-horizon prehensile and non-prehensile manipulation. Li et al. (2024b) use model-based motion planners in high-fidelity simulation to generate trajectories for bimanual contact-rich manipulation, then train task-conditioned diffusion policies on the resulting data. Across this family the learner runs on its own at execution time; TAMP plays no part beyond generating the data.

Distillation via reward shaping. A third family distills planning knowledge into reinforcement learning rather than imitation learning. Grzes and Kudenko (2008) are the earliest example: a STRIPS plan is converted into a potential function for potential-based reward shaping, accelerating RL on a grid-world domain. Schubert et al. (2021) extend the idea to robot manipulation with final-volume-preserving reward shaping, a relaxation of potential-based shaping that admits more aggressive shaping functions while preserving long-term behavior. The learner here is a standard RL agent; the distilled product of TAMP is the reward signal rather than a trajectory.

6.2 Scaffolding

Whereas distillation replaces TAMP at execution time, *scaffolding* retains the planner and uses learning to extend what it can do. TAMP's long-horizon decomposition, model-based reasoning, and skill sequencing remain in place; the learner fills in pieces classical TAMP cannot supply—contact-rich skills, robust low-level controllers, or recovery behaviors. The methods here are *interleaved* in the sense of Section 3: training invokes the planner, and the learned components integrate with TAMP's existing structure at deployment. We identify three families, distinguished by what TAMP supplies to the learning process.

Demonstrations from TAMP. A first family uses the planner to generate or gate training trajectories that the learner consumes offline. Mandlekar et al. (2023) hand off control between a TAMP solver and a human teleoperator, with TAMP handling segments it can solve and the human taking over the contact-rich parts; a single operator can oversee a fleet of robots asynchronously, and the resulting demonstrations train IL policies for the segments TAMP cannot. Zhou et al. (2024) extend HITL-TAMP with sparse-reward RL fine-tuning of the BC-learned skill policies, warm-started from BC and KL-constrained to stay close to it. Garrett et al. (2024) replace teleoperation with motion-planned skill recombination: object-relative reference segments from a few human demonstrations are transformed to new contexts via motion planning, generating many more demonstrations from few inputs. Li et al. (2026) extend this to bimanual mobile manipulation by formulating data generation as a constrained optimization over base reachability and visibility. Watanabe et al. (2023, 2025) use a sampling-based TAMP planner to autonomously generate demonstrations of long-horizon skills, then apply offline RL to extract robust policies that replace the original scripted skills in the library. Curtis et al. (2020) embed a curiosity score in the planner's own sampler and train an imitation-learning policy on the resulting trajectories.

Structural primitives from TAMP. A second family uses TAMP's operator structure as the scaffolding for hierarchical reinforcement learning. The planner provides a decomposition; RL fills in the option policy for each operator. Lyu et al. (2019) couple a symbolic planner with deep RL: each operator becomes an option with initiation and termination sets derived from preconditions and effects, intrinsic rewards push option policies to completion, and a meta-controller updates operator costs from observed success ratios (Jiang et al. 2018). Kokel et al. (2021) extend this with relational state abstractions—first-order conditional influence statements—that reduce the per-option MDP and enable cross-task transfer. Cheng and Xu (2023) integrate the loop further: the task planner generates plans that practice skills needing additional training, and skills are learned over a reduced state space derived from operator arguments. Illanes et al. (2020) frame the same coupling theoretically, providing sufficient conditions under which optimal policies are guaranteed to reach goal states and a dominance ordering among optimal flat, hierarchical, partial-order, and plan-constrained policies. Rather than learning a policy for each given operator, Liu et al. (2026) use model-free RL to discover new options that shortcut the abstract planning graph, including dynamic behaviors such as slapping and wiping that are absent from the hand-specified set.

TAMP as interaction partner. A third family runs the learner and TAMP together at execution time, each

handling what the other cannot. Yamada et al. (2021) extend an RL agent's action space to include calls to a motion planner—large action magnitudes invoke the planner for collision-free transit, small ones execute directly for contact-rich manipulation, with the rescaled action space balancing the two. Vats et al. (2025a) train hierarchical RL policies that activate on failure detection and either complete the task with primitive actions or hand control back to nominal model-based controllers, with the planner providing the nominal options. In both, TAMP is neither replaced nor used only offline: the learner decides when to invoke the planner and when to act on its own.

6.3 Active Learning

The prior two subsections train new components from TAMP-derived signals. *Active learning* takes the components as given and uses planning to decide what data to collect next. The methods here are *interleaved* in the sense of Section 3: the planner runs throughout learning to decide what to attempt or what to ask next. We identify two families, distinguished by whom the active learner queries—the environment via targeted practice or experimentation, or a human teacher via labels and feedback.

Planning to practice or probe. A first family uses the planner to allocate environment interactions among existing components. As antecedent work without a planner, Da Silva et al. (2014) address active task selection when learning a single parameterized skill by estimating the expected improvement in skill performance (over a distribution of tasks) that would result from practicing on a candidate task. Kumar et al. (2024) consider skill parameter samplers: each skill's competence—its expected success rate under its current sampler—is tracked over practice attempts and extrapolated via a Beta-Bernoulli model, and competence-aware planning, with operator costs derived from competence, scores each skill's expected impact on task success so the agent practices the maximally helpful one. Vats et al. (2023) apply the same idea to recovery policies: failure modes are discovered in simulation, and a model-predictive meta-reasoner allocates training resources across candidate recovery skills based on their predicted contribution to task performance. Wang et al. (2018) target operator constraints rather than skill parameters: a Gaussian-process model of constraint feasibility is actively refined by sampling near the level set, with the planner identifying which constraint regions are most relevant to the task distribution. Across this family, the planner uses an internal estimate of skill or model quality to decide where the next interaction will help most.

Planning to query a human teacher. A second family substitutes a human teacher for the environment. The planner decides when to query the human instead of acting in the environment, and what to ask. Kulick et al. (2013) formalize this directly: the robot uses planning to construct pick-and-place configurations that maximize information gain about a grounded relational symbol the human is teaching. Li and Silver (2023) extend the idea to predicate classifiers in bilevel planning: an ensemble of classifiers produces uncertainty estimates, the agent uses the current planning model to find lookahead trajectories that visit high-entropy abstract states, and queries the human for ground-atom labels when local entropy exceeds a threshold. Vats et al. (2025b) generalize the query problem across a sequence of tasks: the agent chooses among skill queries, preference queries, and direct help requests, with the choice formulated as an uncapacitated facility location problem that minimizes cumulative human effort. In both families, the agent plans how to spend a limited budget of environment interactions or human queries.

A common theme across these three subsections is that distillation, scaffolding, and active learning describe distinct integration patterns rather than competing approaches: a system could in principle distill some skills, scaffold others, and actively refine the rest. The choice of pattern reflects where TAMP supplies the most value at deployment—wholly replaced, retained as a runtime host, or used only to direct what to learn—and where learning fills gaps that classical TAMP cannot. Few systems combine multiple modes, and how to compose them remains largely open.

7 The Future of TAMP Learning

The preceding sections traced the three families of TAMP learning (learning to make TAMP possible, learning to make TAMP fast, and using TAMP for learning), and the body of work each has produced. We close with a forward-looking discussion: clear opportunities revealed by the analysis, open challenges that remain, and the role of TAMP learning in an era when foundation models are reshaping robotics.

7.1 Clear Opportunities

The taxonomy of Section 3 and the per-section analyses expose several cells the field has populated thinly. Each suggests a tractable direction for new work.

Guarantee-preserving learning. The methods that learn TAMP *models*, including predicates (Section 4.1), operators (Section 4.3), skill policies (Section 4.2), and policies distilled from TAMP (Section 6.1), almost uniformly abandon TAMP's classical guarantees of soundness, probabilistic completeness, and asymptotic optimality, with a few notable exceptions (Konidaris et al. 2018; Shah and Srivastava 2024). The methods that learn to accelerate planning (Section 5) preserve completeness more often, when they only reorder the planner's search. Extending preservation to

learned models is fundamentally nontrivial: the model spaces are open-ended and high-dimensional, and the inductive biases that make them tractable to learn rarely admit the closed-form analysis classical TAMP guarantees rely on. The Safe Reinforcement Learning literature (García and Fernández 2015) offers partial answers that have not been combined with TAMP-style structure, leaving open directions such as skill policies with bounded competence estimates the planner can incorporate (Kumar et al. 2024).

Native representations in distillation. Distillation (Section 6.1) is, at present, entirely latent: every method produces a neural policy from TAMP-generated supervision. Yet TAMP's own outputs (plans, skeletons, parameter bindings) are highly structured. Distilling TAMP solutions into native representations (decision trees, rule sets, programs, generalized plans) would yield interpretable artifacts that could be audited, edited by hand, and composed with TAMP at deployment. Curtis et al. (2022) learn generalized plans for TAMP speedup, but the analogous program for end-to-end policy distillation is unexplored.

Interleaved acceleration learning. The methods of Section 5 are largely decoupled: planner traces are collected from easier problems, models are trained offline, and the trained models are deployed on harder problems. Active learning of acceleration models, where the planner is run on a target problem distribution and instances are adaptively selected to reveal where future planning will fail, remains underexplored. The active-learning machinery of Section 6.3 could be redirected toward this purpose.

Cross-family integration. No system in the survey spans all three families simultaneously. The holy grail for TAMP learning would be a single agent that accumulates components from data (Section 4), accelerates its own planning as it does so (Section 5), and uses planning to teach and refine downstream learners (Section 6), improving over a lifetime of deployment along all three axes at once. Existing work approaches each family separately; combining them remains an open problem. The necessary components already exist in the methods surveyed here, and in principle could be combined into such a system; in practice, friction at the join points, where outputs of one family must serve as inputs to another, will surface a new family of research problems.

7.2 Open Challenges

The opportunities above are tractable in the sense that the analysis points to plausible recipes. Other questions are harder. We highlight three that have surfaced across the survey but lack settled answers.

Robustness under model error. TAMP planners are brittle to model error: a slight misestimation of geometry, friction, or perception can cause a sound plan to fail catastrophically. The learned components introduced in Sections 4–6 compound this problem by introducing additional model error of their own. Vats et al. (2025a) address recovery from failures, Huang et al. (2025d) leverage failures as informative signals to generate additional data to recover from current failures and minimize future failures, and Li et al. (2024a) learn bridge policies for novelty, but the broader question of how to design TAMP learning systems that degrade gracefully under model error rather than catastrophically remains open. Bounded-uncertainty planning, fallback hierarchies, and contract-based composition of learned and engineered components are partial answers in adjacent fields and have not been productively combined for TAMP learning.

Partial observability and active perception. Most of the survey assumes that the agent can estimate its state precisely enough for planning. Real deployments cannot. Huang et al. (2024b) learn relational dynamics from partial-view point clouds and leverage the predictions to maintain memory about previously observed but currently occluded objects. However, such approaches primarily cope with partial observability through prediction and memory rather than actively gathering information. Active perception, where the agent learns when to look, where to look, and what to model, is a natural application for TAMP learning machinery that the field has barely begun to exploit.

Humans beyond demonstrations. Human input in the survey is dominated by demonstrations: teleoperation traces (Section 6.2), atom labels (Section 6.3), or language prompts (Section 4.1). The space of richer human-robot interaction (preferences, corrections, explanations of intent, mixed-initiative collaboration) is underused. Vats et al. (2025b) formalize query allocation, but how to integrate human input across the full TAMP learning lifecycle remains open. Adjacent fields, including interactive machine learning, learning from demonstration, and human-robot interaction, offer techniques for each of these modes that have not been productively adapted to TAMP learning.

7.3 TAMP Learning in the FM Era

Foundation models are reshaping robotics. A natural question is whether TAMP learning continues to matter as foundation models become more capable. We argue that it does, and that its role is becoming more important rather than less.

Complementary strengths. The competences of foundation models and classical TAMP are largely non-overlapping. Foundation models bring open-vocabulary perception, common-sense knowledge, semantic generalization, and natural-language interfaces, capabilities that classical TAMP has historically assumed as inputs or delegated to other modules. TAMP provides long-horizon sequential reasoning under constraints, certified completeness, and deployment to domains where large-scale pretraining data does not exist. The

two are now beginning to be combined directly: TidyBot++ (Shen et al. 2026), for example, coordinates a GPU-parallelized TAMP solver with pretrained models for object detection, language grounding, stereo depth, grasping, and segmentation to solve long-horizon manipulation tasks from raw pixels and natural-language instructions without robot-specific training.

Components from foundation models. Across all three families of TAMP learning, components that were once hand-engineered or trained from scratch are now routinely supplied by foundation models. Predicates that previously required manual specification or from-scratch classifier training are being proposed by vision-language models or extracted from language feedback (Liang et al. 2025; Athalye et al. 2026; Han et al. 2024; Byrnes et al. 2025). Plan skeletons, abstract action sequences, and high-level decompositions that previously required either an engineered planning model or a learned policy are now generated directly by large language models (Rana et al. 2023; Lin et al. 2023; Dalal et al. 2024; Huang et al. 2025c; Lee et al. 2025a; Kumar et al. 2026). Skill controllers and low-level policies are increasingly composed as vision-language-action models (Shi et al. 2025). Even failure detection and recovery, traditionally hand-coded, are being delegated to vision-language reasoning (Wang et al. 2024; Duan et al. 2025; Yan et al. 2025). TAMP learning will continue to benefit from improvements in foundation models, though training from scratch will remain essential where pretraining data is scarce.

The importance of certification. The capabilities of foundation models also amplify the case for certifiable structure around them. A purely VLA-driven robot operating in a home, a hospital, or a warehouse needs guarantees of safety and goal achievement that current foundation models cannot supply on their own. TAMP's tradition of formal guarantees becomes more relevant, not less, when the components inside the planner are more powerful but more opaque. The guarantee-preserving learning highlighted in Section 7.1 is doubly motivated in this context: it is the bridge between web-scale capability and deployable certification.

TAMP learning for foundation models. The reciprocal opportunity is also present: TAMP could increasingly provide training signals for foundation models, not just consume them. The patterns surveyed in Section 6 (distillation, scaffolding, and active learning) apply equally when the downstream learner is a vision-language-action model rather than a bespoke policy. TAMP-generated demonstrations could post-train VLAs to specialize in long-tail tasks; TAMP-derived reward signals could fine-tune foundation models on goal-directed behavior; TAMP-driven query selection could focus data collection on regions of the task distribution where pretraining is sparsest. This reciprocity positions TAMP learning not only as a consumer of foundation models but as a tool for shaping them.

Acknowledgments

This work was partially supported by a Princeton SEAS Innovation Grant, by ONR under grant 00014-22-1-2592, and by NSF IIS #2419809. Disclosure: George Konidaris is the Chief Scientific Advisor of Realtime Robotics, a robotics company that develops motion planning software.

References

Abbatematteo B, Rosen E, Thompson S, Akbulut T, Rammohan S and Konidaris G (2024) Composable interaction primitives: A structured policy class for efficiently learning sustained-contact manipulation skills. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 7522–7529.

Abraham I, De La Torre G and Murphey TD (2017) Model-based control using koopman operators. In: *Robotics: Science and Systems (RSS)*.

Achterhold J, Krimmel M and Stueckler J (2023) Learning temporally extended skills in continuous domains as symbolic actions for planning. In: *Conference on Robot Learning (CoRL)*. pp. 225–236.

Agia C, Migimatsu T, Wu J and Bohg J (2023) STAP: Sequencing Task-Agnostic Policies. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 7951–7958.

Ahmed Z, Tenenbaum JB, Bates C and Gershman SJ (2025) Synthesizing world models for bilevel planning. *Transactions on Machine Learning Research (TMLR)* .

Ahmetoglu A, Seker MY, Piater J, Oztop E and Ugur E (2022) DeepSym: Deep Symbol Generation and Rule Learning for Planning from Unsupervised Robot Interaction. *Journal of Artificial Intelligence Research (JAIR)* 75: 709–745.

Allen KR, Lopez-Guevara T, Rubanova Y, Stachenfeld KL, Sanchez-Gonzalez A, Battaglia PW and Pfaff T (2023) Graph network simulators can learn discontinuous, rigid contact dynamics. In: *Conference on Robot Learning (CoRL)*.

Ames B, Thackston A and Konidaris G (2018) Learning Symbolic Representations for Planning with Parameterized Skills. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 526–533.

Andersen G and Konidaris G (2017) Active exploration for learning symbolic representations. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Arora A, Fiorino H, Pellier D, Etivier MM and Pesty S (2018) A Review of Learning Planning Action Models. *The Knowledge Engineering Review* 33.

Asai M, Kajino H, Fukunaga A and Muise C (2022) Classical Planning in Deep Latent Space. *Journal of Artificial Intelligence Research (JAIR)* 74: 1599–1686.

Assran M, Bardes A, Fan D, Garrido Q, Howes R, Komeili M, Muckley M, Rizvi A, Roberts C, Sinha K, Zholus A, Arnaud S, Gejji A, Martin A, Robert Hogan F, Dugas D, Bojanowski P, Khalidov V, Labatut P, Massa

F, Szafraniec M, Krishnakumar K, Li Y, Ma X, Chandar S, Meier F, LeCun Y, Rabbat M and Ballas N (2025) V-JEPA 2: Self-supervised video models enable understanding, prediction and planning.

Athalye A, Kumar N, Silver T, Liang Y, Wang J, Lozano-Pérez T and Kaelbling LP (2026) From Pixels to Predicates: Learning Symbolic World Models Via Pretrained VLMs. *IEEE Robotics and Automation Letters (RA-L)* : 1–8.

Bacchus F and Yang Q (1994) Downward refinement and the efficiency of hierarchical problem solving. *Artificial Intelligence (AIJ)* 71(1): 43–100.

Bagaria A and Konidaris G (2020) Option Discovery using Deep Skill Chaining. In: *International Conference on Learning Representations (ICLR)*.

Barto AG and Mahadevan S (2003) Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems* 13(4): 341–379. DOI:10.1023/A:1025696116075.

Bonet B and Geffner H (2001) Planning as heuristic search. *Artificial Intelligence* 129(1): 5–33.

Bouhsain SA, Alami R and Siméon T (2023) Learning to Predict Action Feasibility for Task and Motion Planning in 3D Environments. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 3736–3742.

Bradley C and Roy N (2022) Learning to Guide Search in Long-Horizon Task and Motion Planning. In: *CoRL 2022 Workshop on Long-Horizon Planning*.

Byrnes W, Bogdanovic M, Balakirsky A, Balakirsky S and Garg A (2025) CLIMB: Language-guided continual learning for task planning with iterative model building. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Cambon S, Alami R and Gravot F (2009) A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research (IJRR)* 28(1): 104–126.

Chen W, Huang J, Pang O, Li Z, Hu X, Zhang L, Zhang Z, Coates M, Cao T, Quan X and Zhang Y (2026) H-wm: Robotic task and motion planning guided by hierarchical world model. *arXiv preprint arXiv:2602.11291* .

Cheng S and Xu D (2023) LEAGUE: Guided Skill Learning and Abstraction for Long-Horizon Manipulation. *IEEE Robotics and Automation Letters (RA-L)* 8(10): 6451–6458.

Chitnis R, Hadfield-Menell D, Gupta A, Srivastava S, Groshev E, Lin C and Abbeel P (2016) Guided search for task and motion plans using learned heuristics. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 447–454.

Chitnis R, Kaelbling LP and Lozano-Pérez T (2019) Learning quickly to plan quickly using modular meta-learning. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Chitnis R, Silver T, Kim B, Kaelbling L and Lozano-Perez T (2021a) CAMPs: Learning Context-Specific Abstractions for Efficient Planning in Factored MDPs. In: *Conference on Robot Learning (CoRL)*. pp. 64–79.

Chitnis R, Silver T, Tenenbaum JB, Kaelbling LP and Lozano-Pérez T (2021b) GLIB: Efficient Exploration for Relational Model-Based Reinforcement Learning via Goal-Literal Babbling. *AAAI Conference on Artificial Intelligence (AAAI)* 35(13): 11782–11791.

Chitnis R, Silver T, Tenenbaum JB, Lozano-Pérez T and Kaelbling LP (2022) Learning Neuro-Symbolic Relational Transition Models for Bilevel Planning. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 4166–4173.

Cieślar B, Kaelbling LP, Lozano-Pérez T and Mendez-Mendez J (2025) Learning long-horizon action dependencies in sampling-based bilevel planning. In: Agrawal P, Kroemer O and Burgard W (eds.) *Conference on Robot Learning (CoRL)*, volume 270. pp. 4235–4252.

Curtis A, Kumar N, Cao J, Lozano-Pérez T and Kaelbling LP (2025) Trust the PRoC3S: Solving Long-Horizon Robotics Problems with LLMs and Constraint Satisfaction. In: *Conference on Robot Learning (CoRL)*. pp. 1362–1383.

Curtis A, Silver T, Tenenbaum JB, Lozano-Pérez T and Kaelbling L (2022) Discovering State and Action Abstractions for Generalized Task and Motion Planning. *AAAI Conference on Artificial Intelligence (AAAI)* 36(5): 5377–5384.

Curtis A, Xin M, Arumugam D, Feigelis K and Yamins D (2020) Flexible and Efficient Long-Range Planning Through Curious Exploration. In: *International Conference on Machine Learning (ICML)*. pp. 2238–2249.

Da Silva B, Konidaris G and Barto A (2012) Learning parameterized skills. In: *International Conference on Machine Learning (ICML)*.

Da Silva B, Konidaris G and Barto A (2014) Active learning of parameterized skills. In: Xing EP and Jebara T (eds.) *Proceedings of the 31st International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, volume 32. Bejing, China: PMLR, pp. 1737–1745.

Dalal M, Chiruvolu T, Chaplot D and Salakhutdinov R (2024) Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks. In: *International Conference on Learning Representations (ICLR)*.

Dalal M, Mandlekar A, Garrett CR, Handa A, Salakhutdinov R and Fox D (2023) Imitating task and motion planning with visuomotor transformers. In: Tan J, Toussaint M and Darvish K (eds.) *Conference on Robot Learning (CoRL)*, volume 229. pp. 2565–2593.

Dantam NT, Kingston ZK, Chaudhuri S and Kavraki LE (2018) An incremental constraint-based framework for task and motion planning. *The International Journal of Robotics Research (IJRR)* 37(10): 1134–1151.

Dhakal R, Nguyen DM, Silver T, Xiao X and Stein GJ (2026) Anticipatory Task and Motion Planning: Improved Rearrangement in Persistent Continuous-Space Environments. *IEEE Robotics and Automation*

Letters (RA-L) 11(2): 1850–1857.

Diehl M, Paxton C and Ramirez-Amaro K (2021) Automated Generation of Robotic Planning Domains from Observations. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 6732–6738. ISSN: 2153-0866.

Dittadi A, Drachmann FK and Bolander T (2021) Planning from pixels in atari with learned symbolic representations. In: *AAAI Conference on Artificial Intelligence (AAAI)*, volume 35.

Driess D, Ha JS, Tedrake R and Toussaint M (2021a) Learning Geometric Reasoning and Control for Long-Horizon Tasks from Visual Input. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 14298–14305.

Driess D, Ha JS and Toussaint M (2021b) Learning to solve sequential physical reasoning problems from a scene image. *The International Journal of Robotics Research (IJRR)* 40(12-14): 1435–1466.

Driess D, Oguz O, Ha JS and Toussaint M (2020) Deep Visual Heuristics: Learning Feasibility of Mixed-Integer Programs for Manipulation Planning. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 9563–9569.

Duan J, Pumacay W, Kumar N, Wang YR, Tian S, Yuan W, Krishna R, Fox D, Mandlekar A and Guo Y (2025) AHA: A Vision-Language-Model for Detecting and Reasoning Over Failures in Robotic Manipulation. In: *International Conference on Learning Representations (ICLR)*.

Fang X, Garrett CR, Eppner C, Lozano-Pérez T, Kaelbling LP and Fox D (2024) DiMSam: Diffusion Models as Samplers for Task and Motion Planning under Partial Observability. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 1412–1419.

García J and Fernández F (2015) A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research (JMLR)* 16(1): 1437–1480.

Garrett C, Mandlekar A, Wen B and Fox D (2024) Skillmimicgen: Automated demonstration generation for efficient skill learning and deployment. In: *Conference on Robot Learning (CoRL)*.

Garrett CR, Chitnis R, Holladay R, Kim B, Silver T, Kaelbling LP and Lozano-Pérez T (2021) Integrated Task and Motion Planning. *Annual Review of Control, Robotics, and Autonomous Systems* 4(1): 265–293.

Garrett CR, Lozano-Pérez T and Kaelbling LP (2020) PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning. *International Conference on Automated Planning and Scheduling (ICAPS)* 30: 440–448.

Ghallab M, Nau D and Traverso P (2025) *Acting, Planning, and Learning*. Cambridge University Press.

Grzes M and Kudenko D (2008) Plan-based reward shaping for reinforcement learning. In: *IEEE International Conference on Intelligent Systems (IS)*, volume 2. pp. 10–22–10–29.

Guo H, Wu F, Qin Y, Li R, Li K and Li K (2023) Recent Trends in Task and Motion Planning for Robotics: A Survey. *ACM Computing Surveys* 55(13s): 289:1–289:36.

Hafner D, Lillicrap T, Norouzi M and Ba J (2021) Mastering Atari with discrete world models. In: *International Conference on Learning Representations (ICLR)*.

Han M, Zhu Y, Zhu SC, Wu YN and Zhu Y (2024) INTERPRET: Interactive Predicate Learning from Language Feedback for Generalizable Task Planning. In: *Robotics: Science and Systems (RSS)*, volume 20.

Han Y, Xie M, Zhao Y and Ravichandar H (2023) On the utility of koopman operator theory in learning dexterous manipulation skills. In: *Conference on Robot Learning (CoRL)*. pp. 106–126.

Hauser K and Latombe JC (2010) Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research (IJRR)* 29(7): 897–915.

Helmert M (2006) The fast downward planning system. *Journal of Artificial Intelligence Research (JAIR)* 26: 191–246.

Hoffmann J and Nebel B (2001) The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 14: 253–302.

Huang J, Tao A, Marco R, Bogdanovic M, Kelly J and Shkurti F (2025a) Automated Planning Domain Inference for Task and Motion Planning. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 12534–12540.

Huang W, Abbeel P, Pathak D and Mordatch I (2022) Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In: *International Conference on Machine Learning (ICML)*. pp. 9118–9147.

Huang W, Chao YW, Mousavian A, Liu MY, Fox D, Mo K and Fei-Fei L (2026) Pointworld: Scaling 3d world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782* .

Huang W, Wang C, Li Y, Zhang R and Fei-Fei L (2025b) ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation. In: *Conference on Robot Learning (CoRL)*. pp. 4573–4602.

Huang W, Wang C, Zhang R, Li Y, Wu J and Fei-Fei L (2023) Voxposer: Composable 3d value maps for robotic manipulation with language models. In: *Conference on Robot Learning (CoRL)*.

Huang Y, Agia C, Wu J, Hermans T and Bohg J (2025c) Points2Plans: From Point Clouds to Long-Horizon Plans with Composable Relational Dynamics. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 1208–1216.

Huang Y, Alvina N, Devendran Shanthi M and Hermans T (2025d) Fail2progress: Learning from real-world robot failures with stein variational inference. In: *Conference on Robot Learning (CoRL)*.

Huang Y, Taylor NC, Conkey A, Liu W and Hermans T (2024a) Latent Space Planning for Multi-Object Manipulation with Environment-Aware Relational Classifiers. *IEEE Transactions on Robotics (T-RO)* .

Huang Y, Yuan J, Kim C, Pradhan P, Chen B, Fuxin L and Hermans T (2024b) Out of Sight, Still in Mind: Reasoning and Planning about Unobserved Objects with Video Tracking Enabled Memory Models. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Illanes L, Yan X, Icarte RT and McIlraith SA (2020) Symbolic Plans as High-Level Instructions for Reinforcement Learning. *International Conference on Automated Planning and Scheduling (ICAPS)* 30: 540–550.

James S, Rosman B and Konidaris G (2022) Autonomous learning of object-centric abstractions for high-level planning. In: *International Conference on Learning Representations (ICLR)*.

Jetchev N, Lang T and Toussaint M (2013) Learning Grounded Relational Symbols from Continuous Data for Abstract Reasoning. In: *Proceedings of the 2013 ICRA Workshop on Autonomous Learning*.

Jiang Y, Yang F, Zhang S and Stone P (2018) Integrating Task-Motion Planning with Reinforcement Learning for Robust Decision Making in Mobile Robots. ArXiv:1811.08955 [cs].

Jung H, Lee D, Park H, Kim J and Kim B (2025) SPIN: distilling Skill-RRT for long-horizon prehensile and non-prehensile manipulation. In: *Conference on Robot Learning (CoRL)*.

Kaelbling LP and Lozano-Pérez T (2011) Hierarchical task and motion planning in the now. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 1470–1477.

Kavraki L, Svestka P, Latombe JC and Overmars M (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4): 566–580.

Keller L, Tanneberg D and Peters J (2025) Neuro-Symbolic Imitation Learning: Discovering Symbolic Abstractions for Skill Learning. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 6519–6526.

Khodeir M, Agro B and Shkurti F (2023a) Learning to Search in Task and Motion Planning With Streams. *IEEE Robotics and Automation Letters (RA-L)* 8(4): 1983–1990.

Khodeir M, Sonwane A, Hari R and Shkurti F (2023b) Policy-Guided Lazy Search with Feedback for Task and Motion Planning. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 3743–3749.

Kim B, Kaelbling L and Lozano-Pérez T (2018) Guiding Search in Continuous State-Action Spaces by Learning an Action Sampler From Off-Target Search Experience. *AAAI Conference on Artificial Intelligence (AAAI)* 32(1).

Kim B, Shimanuki L, Kaelbling LP and Lozano-Pérez T (2022) Representation, learning, and planning algorithms for geometric task and motion planning. *The International Journal of Robotics Research (IJRR)* 41(2): 210–231.

Kim B, Wang Z, Kaelbling LP and Lozano-Pérez T (2019) Learning to guide task and motion planning using score-space representation. *The International Journal of Robotics Research (IJRR)* 38(7): 793–812.

Klissarov M, Bagaria A, Luo Z, Konidaris G, Precup D and Machado MC (2025) Discovering temporal structure: An overview of hierarchical reinforcement learning.

Kokel H, Manoharan A, Natarajan S, Ravindran B and Tadepalli P (2021) RePReL: Integrating Relational Planning and Reinforcement Learning for Effective Abstraction. *International Conference on Automated Planning and Scheduling (ICAPS)* 31: 533–541.

Konidaris G and Barto AG (2009) Skill discovery in continuous reinforcement learning domains using skill chaining. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Konidaris G, Kaelbling LP and Lozano-Perez T (2018) From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning. *Journal of Artificial Intelligence Research (JAIR)* 61: 215–289.

Kress-Gazit H, Fainekos GE and Pappas GJ (2009) Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics* 25(6): 1370–1381. DOI:10.1109/TRO.2009.2030225.

Kulick J, Toussaint M, Lang T and Lopes M (2013) Active learning for teaching a robot grounded relational symbols. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. pp. 1451–1457.

Kumar N, McClinton W, Chitnis R, Silver T, Lozano-Pérez T and Kaelbling LP (2023) Learning Efficient Abstract Planning Models that Choose What to Predict. In: *Conference on Robot Learning (CoRL)*. pp. 2070–2095.

Kumar N, Shen W, Ramos F, Fox D, Lozano-Pérez T, Kaelbling LP and Garrett CR (2026) Open-World Task and Motion Planning via Vision-Language Model Generated Constraints. *IEEE Robotics and Automation Letters (RA-L)* 11(3): 3366–3373.

Kumar N, Silver T, McClinton W, Zhao L, Proulx S, Lozano-Pérez T, Kaelbling LP and Barry JL (2024) Practice Makes Perfect: Planning to Learning Skill Parameter Policies. In: *Robotics: Science and Systems (RSS)*, volume 20.

Kwon M, Kim Y and Kim YJ (2025) Fast and Accurate Task Planning using Neuro-Symbolic Language Models and Multi-Level Goal Decomposition. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 16195–16201.

Kwon M and Kim YJ (2026) Kinodynamic Task and Motion Planning using VLM-guided and Interleaved Sampling. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

LaValle SM and James J Kuffner, Jr (2001) Randomized kinodynamic planning. *The International Journal of Robotics Research (IJRR)* 20(5): 378–400.

Lee D, Joo S, Lee K and Kim B (2025a) Prime the search: Using large language models for guiding geometric task and motion planning by warm-starting tree search. *The International Journal of Robotics Research (IJRR)* : 02783649251347307.

Lee Y, Li AZ, Huang P, Heiden E, Jatavallabhula KM, Damken F, Smith K, Nowrouzezahrai D, Ramos F and Shkurti F (2025b) STAMP: Differentiable Task and Motion Planning via Stein Variational Gradient Descent. *IEEE Robotics and Automation Letters (RA-L)* 10(6): 6007–6014.

Li A, Kumar N, Lozano-Pérez T and Kaelbling LP (2024a) Learning to Bridge the Gap: Efficient Novelty Recovery with Planning and Reinforcement Learning.

Li A and Silver T (2023) Embodied Active Learning of Relational State Abstractions for Bilevel Planning. In: *Conference on Lifelong Learning Agents (CoLLAs)*. pp. 358–375.

Li AK, Silva TC, Edwards V, Kumar V and Hsieh MA (2025a) Koopmotion: Learning almost divergence free koopman flow fields for motion planning. In: *Conference on Robot Learning (CoRL)*.

Li B, Silver T, Scherer S and Gray A (2025b) Bilevel Learning for Bilevel Planning. In: *Robotics: Science and Systems (RSS)*.

Li C, Xu M, Bahety A, Yin H, Jiang Y, Huang H, Wong J, Garlanka S, Gokmen C, Zhang R, Liu W, Wu J, Martín-Martín R and Li FF (2026) MoMaGen: Generating Demonstrations under Soft and Hard Constraints for Multi-Step Bimanual Mobile Manipulation. In: *International Conference on Learning Representations (ICLR)*.

Li L, Walsh TJ and Littman ML (2006) Towards a unified theory of state abstraction for MDPs. In: *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*.

Li X, Zhao T, Zhu X, Wang J, Pang T and Fang K (2024b) Planning-Guided Diffusion Policy Learning for Generalizable Contact-Rich Bimanual Manipulation. ArXiv:2412.02676 [cs.RO].

Liang Y, Kumar N, Tang H, Weller A, Tenenbaum JB, Silver T, Henriques JF and Ellis K (2025) VisualPredicator: Learning Abstract World Models with Neuro-Symbolic Predicates for Robot Planning. In: *International Conference on Learning Representations (ICLR)*.

Liang Y, Nguyen D, Yang C, Li T, Tenenbaum JB, Rasmussen CE, Weller A, Tavares Z, Silver T and Ellis K (2026) ExoPredicator: Learning Abstract Models of Dynamic Worlds for Robot Planning. In: *International Conference on Learning Representations (ICLR)*.

Lin K, Agia C, Migimatsu T, Pavone M and Bohg J (2023) Text2Motion: from natural language instructions to feasible plans. *Autonomous Robots* 47(8): 1345–1365.

Liu G, de Winter J, Durodié Y, Steckelmacher D, Nowe A and Vanderborght B (2024a) Optimistic Reinforcement Learning-Based Skill Insertions for Task and Motion Planning. *IEEE Robotics and Automation Letters (RA-L)* 9(6): 5974–5981.

Liu W, Nie N, Zhang R, Mao J and Wu J (2024b) Learning compositional behaviors from demonstration and language. In: *Conference on Robot Learning (CoRL)*.

Liu YI, Li B, Eysenbach B and Silver T (2026) SLAP: Shortcut Learning for Abstract Planning. In: *International Conference on Learning Representations (ICLR)*.

Lorang P, Lu H, Huemer J, Zips P and Scheutz M (2025a) Few-Shot Neuro-Symbolic Imitation Learning for Long-Horizon Planning and Acting. In: *Conference on Robot Learning (CoRL)*. pp. 2501–2518.

Lorang P, Lu H and Scheutz M (2025b) Curiosity-Driven Imagination: Discovering Plan Operators and Learning Associated Policies for Open-World Adaptation. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 12557–12564.

Luong L (2023) *Learning Refinement Cost Estimators for Bilevel Planning*. Master of engineering thesis, Massachusetts Institute of Technology. B.S. Computer Science and Engineering, MIT (2022).

Lyu D, Yang F, Liu B and Gustafson S (2019) SDRL: Interpretable and Data-Efficient Deep Reinforcement Learning Leveraging Symbolic Planning. *AAAI Conference on Artificial Intelligence (AAAI)* 33(01): 2970–2977.

Mandlekar A, Garrett CR, Xu D and Fox D (2023) Human-in-the-Loop Task and Motion Planning for Imitation Learning. In: *Conference on Robot Learning (CoRL)*. pp. 3030–3060.

Mao J, Lozano-Pérez T, Tenenbaum JB and Kaelbling LP (2022) PDSketch: Integrated domain programming, learning, and planning. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Marthi B, Russell SJ and Wolfe JA (2007) Angelic semantics for high-level actions. In: *International Conference on Automated Planning and Scheduling (ICAPS)*.

Mayr M, Ahmad F, Chatzilygeroudis K, Nardi L and Krueger V (2022) Skill-based Multi-objective Reinforcement Learning of Industrial Robot Tasks with Planning and Knowledge Integration. In: *IEEE International Conference on Robotics and Biomimetics (ROBIO)*. pp. 1995–2002.

McDermott D, Ghallab M, Howe AE, Knoblock CA, Ram A, Veloso MM, Weld DS and Wilkins DE (1998) PDDL—The Planning Domain Definition Language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

McDonald MJ and Hadfield-Menell D (2022) Guided Imitation of Task and Motion Planning. In: *Conference on Robot Learning (CoRL)*. pp. 630–640.

Mendez-Mendez J, Kaelbling LP and Lozano-Pérez T (2023) Embodied Lifelong Learning for Task and Motion Planning. In: *Conference on Robot Learning (CoRL)*.

Migimatsu T and Bohg J (2022) Grounding predicates through actions. In: *IEEE International Conference on*

Robotics and Automation (ICRA).

Mishra UA, Xue S, Chen Y and Xu D (2023) Generative skill chaining: Long-horizon skill planning with diffusion models. In: Tan J, Toussaint M and Darvish K (eds.) *Conference on Robot Learning (CoRL)*, volume 229. pp. 2905–2925.

Moerland TM, Broekens J, Plaat A and Jonker CM (2023) Model-based reinforcement learning: A survey. *Found. Trends Mach. Learn.* 16(1): 1–118. DOI:10.1561/2200000086.

Moreno MD, García N, Gómez V and Geffner H (2024) Combined Task and Motion Planning via Sketch Decompositions. *International Conference on Automated Planning and Scheduling (ICAPS)* 34: 123–132.

Noseworthy M, Moses C, Brand I, Castro S, Kaelbling L, Lozano-Pérez T and Roy N (2021) Active Learning of Abstract Plan Feasibility. In: *Robotics: Science and Systems (RSS)*. ArXiv:2107.00683 [cs].

Ortiz-Haro J, Ha JS, Driess D and Toussaint M (2022) Structured deep generative models for sampling on constraint manifolds in sequential manipulation. In: *Conference on Robot Learning (CoRL)*. pp. 213–223.

Pacheck A, James S, Konidaris G and Kress-Gazit H (2023) Automatic encoding and repair of reactive high-level tasks with learned abstract representations. *The International Journal of Robotics Research (IJRR)* 42(4-5): 263–288.

Pasula HM, Zettlemoyer LS and Kaelbling LP (2007) Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research (JAIR)* 29: 309–352.

Pateria S, Subagdja B, Tan Ah and Quek C (2021) Hierarchical reinforcement learning: A comprehensive survey. *ACM Comput. Surv.* 54(5). DOI:10.1145/3453160.

Paxton C, Raman V, Hager GD and Kobilarov M (2017) Combining neural networks and tree search for task and motion planning in challenging environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 6059–6066.

Plaku E and Karaman S (2016) Motion planning with temporal-logic specifications: Progress and challenges. *AI Communications* 29(1): 151–162. DOI:10.3233/AIC-150682. URL https://journals.sagepub.com/doi/abs/10.3233/AIC-150682.

Rana K, Haviland J, Garg S, Abou-Chakra J, Reid I and Suenderhauf N (2023) SayPlan: Grounding large language models using 3d scene graphs for scalable robot task planning. In: *Conference on Robot Learning (CoRL)*.

Ren T, Cowen-Rivers AI, Ammar HB and Peters J (2022) Learning Geometric Constraints in Task and Motion Planning. ArXiv:2201.09612 [cs].

Sarathy V, Kasenberg D, Goel S, Sinapov J and Scheutz M (2021) SPOTTER: Extending Symbolic Planning Operators through Targeted Reinforcement Learning. In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. pp. 1118–1126.

Schrittwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, Guez A, Lockhart E, Hassabis D, Graepel T, Lillicrap T and Silver D (2020) Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* 588: 604–609.

Schubert I, Oguz OS and Toussaint M (2021) Plan-Based Relaxed Reward Shaping for Goal-Directed Tasks. In: *International Conference on Learning Representations (ICLR)*.

Shah N, Nagpal J and Srivastava S (2025) From Real World to Logic and Back: Learning Generalizable Relational Concepts For Long Horizon Robot Planning. In: *Conference on Robot Learning (CoRL)*. pp. 5362–5434.

Shah N and Srivastava S (2024) Hierarchical Planning and Learning for Robots in Stochastic Settings Using Zero-Shot Option Invention. In: *AAAI Conference on Artificial Intelligence (AAAI)*, volume 38. pp. 10358–10367.

Shao YS, Zheng Y, Sun S, Chaudhari P, Kumar V and Figueroa N (2026) Symskill: Symbol and Skill Co-Invention for Data-Efficient and Real-Time Long-Horizon Manipulation. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Shen W, Garrett CR, Kumar N, Goyal A, Hermans T, Kaelbling LP, Lozano-Pérez T and Ramos F (2025) Differentiable gpu-parallelized task and motion planning. In: *Robotics: Science and Systems (RSS)*.

Shen W, Kumar N, Chintalapudi S, Wang J, Watson C, Hu ES, Cao J, Jayaraman D, Kaelbling LP and Lozano-Pérez T (2026) TiPToP: A Modular Open-Vocabulary Planning System for Robotic Manipulation. ArXiv:2603.09971 [cs.RO].

Shi LX, Ichter B, Equi M, Ke L, Pertsch K, Vuong Q, Tanner J, Walling A, Wang H, Fusai N, , Li-Bell A, Driess D, Groom L, Levine S and Finn C (2025) Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In: *International Conference on Machine Learning (ICML)*.

Silver T, Athalye A, Tenenbaum JB, Lozano-Pérez T and Kaelbling LP (2022) Learning Neuro-Symbolic Skills for Bilevel Planning. In: *Conference on Robot Learning (CoRL)*.

Silver T, Chitnis R, Curtis A, Tenenbaum JB, Lozano-Pérez T and Kaelbling LP (2021a) Planning with Learned Object Importance in Large Problem Instances using Graph Neural Networks. *AAAI Conference on Artificial Intelligence (AAAI)* 35(13): 11962–11971.

Silver T, Chitnis R, Kumar N, McClinton W, Lozano-Pérez T, Kaelbling L and Tenenbaum JB (2023) Predicate Invention for Bilevel Planning. *AAAI Conference on Artificial Intelligence (AAAI)* 37(10): 12120–12129.

Silver T, Chitnis R, Tenenbaum J, Kaelbling LP and Lozano-Pérez T (2021b) Learning Symbolic Operators for Task and Motion Planning. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 3182–3189.

Siméon T, Laumond JP, Cortés J and Sahbani A (2004) Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research (IJRR)* 23(7-8): 729–746.

Smolensky P (1988) On the proper treatment of connectionism. *Behavioral and Brain Sciences* 11(1): 1–23.

Srivastava S, Fang E, Riano L, Chitnis R, Russell S and Abbeel P (2014) Combined task and motion planning through an extensible planner-independent interface layer. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 639–646.

Sun X and Shoukry Y (2024) Neurosymbolic motion and task planning for linear temporal logic tasks. *IEEE Transactions on Robotics (T-RO)* 40: 2749–2768.

Sung Y, Wang Z and Stone P (2023) Learning to Correct Mistakes: Backjumping in Long-Horizon Task and Motion Planning. In: *Conference on Robot Learning (CoRL)*. pp. 2115–2124.

Sutton RS, Precup D and Singh S (1999) Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence (AIJ)* .

Toussaint M (2015) Logic-geometric programming: An optimization-based approach to combined task and motion planning. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. pp. 1930–1936.

Toussaint M, Allen K, Smith K and Tenenbaum J (2018) Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning. In: *Robotics: Science and Systems (RSS)*.

Ugur E and Piater J (2015) Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 2627–2633.

Urbaniak D, Agostini A and Lee D (2021) Combining Task and Motion Planning using Policy Improvement with Path Integrals. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. pp. 149–155.

van den Oord A, Vinyals O and Kavukcuoglu K (2017) Neural discrete representation learning. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Vats S, Jha DK, Likhachev M, Kroemer O and Romeres D (2025a) RecoveryChaining: Learning Local Recovery Policies for Robust Manipulation. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Vats S, Likhachev M and Kroemer O (2023) Efficient Recovery Learning using Model Predictive Meta-Reasoning. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Vats S, Zhao M, Callaghan P, Jia M, Likhachev M, Kroemer O and Konidaris G (2025b) Optimal interactive learning on the job via facility location planning. In: *Robotics: Science and Systems (RSS)*.

Verma P, Marpally SR and Srivastava S (2022) Discovering User-Interpretable Capabilities of Black-Box Planning Agents. *International Conference on Principles of Knowledge Representation and Reasoning (KR)* 19(1): 362–372. Conference Name: Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning.

Wang S, Han M, Jiao Z, Zhang Z, Wu YN, Zhu SC and Liu H (2024) LLM3: Large Language Model-based Task and Motion Planning with Motion Failure Reasoning. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 12086–12092.

Wang Y, Syed R, Wu F, Zhang M, Onol A, Barreiros J, Nayyeri H, Dear T, Zhang H and Li Y (2026) Interactive world simulator for robot policy training and evaluation. *arXiv preprint arXiv:2603.08546* .

Wang Z, Garrett CR, Kaelbling LP and Lozano-Pérez T (2018) Active Model Learning and Diverse Action Sampling for Task and Motion Planning. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 4107–4114.

Wang Z, Garrett CR, Kaelbling LP and Lozano-Pérez T (2021) Learning compositional models of robot skills for task and motion planning. *The International Journal of Robotics Research (IJRR)* 40(6-7): 866–894.

Watanabe S, Horn G, Tørresen J and Ellefsen KO (2023) Offline Skill Generalization via Task and Motion Planning. ArXiv:2311.14328 [cs].

Watanabe S, Horn G, Tørresen J and Ellefsen KO (2025) Integrating Bilevel Planning and Offline Skill Learning for Enhancing Mobile Manipulation. In: *IEEE International Conference on Automation Science and Engineering (CASE)*. pp. 2275–2280.

Wells AM, Dantam NT, Shrivastava A and Kavraki LE (2019) Learning Feasibility for Task and Motion Planning in Tabletop Environments. *IEEE Robotics and Automation Letters (RA-L)* 4(2): 1255–1262.

Xia V, Wang Z, Allen K, Silver T and Kaelbling LP (2019) Learning sparse relational transition models. In: *International Conference on Learning Representations (ICLR)*.

Xu D, Mandlekar A, Martín-Martín R, Zhu Y, Savarese S and Fei-Fei L (2021) Deep affordance foresight: Planning through what can be done in the future. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Xu L, Ren T, Chalvatzaki G and Peters J (2022) Accelerating Integrated Task and Motion Planning with Neural Feasibility Checking. ArXiv:2203.10568 [cs].

Xue T, Razmjoo A, Shetty S and Calinon S (2024) Logic-Skill Programming: An Optimization-based Approach to Sequential Skill Planning. In: *Robotics: Science and Systems (RSS)*.

Yamada J, Lee Y, Salhotra G, Pertsch K, Pflueger M, Sukhatme G, Lim J and Englert P (2021) Motion planner augmented reinforcement learning for robot manipulation in obstructed environments. In: Kober J, Ramos F and Tomlin C (eds.) *Conference on Robot Learning (CoRL)*, volume 155. pp. 589–603.

Yan M, Mengdibayev M, Floros A, Guo W, Kavraki LE and Kingston Z (2025) Using VLM reasoning to constrain task and motion planning.

Yang F, Lyu D, Liu B and Gustafson S (2018) PEORL: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. pp. 4860–4866.

Yang Z, Garrett C, Fox D, Lozano-Pérez T and Kaelbling LP (2025) Guiding Long-Horizon Task and Motion Planning with Vision Language Models. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 16847–16853.

Yang Z, Garrett CR, Lozano-Perez T, Kaelbling L and Fox D (2023) Sequence-Based Plan Feasibility Prediction for Efficient Task and Motion Planning. In: *Robotics: Science and Systems (RSS)*, volume 19.

Yang Z, Hedegaard B, Jaafar A, Wei Y, Thompson S, Raman SS, Fu H, Tellex S, Konidaris G, Paulius D and Shah N (2026) SkillWrapper: Generative Predicate Invention for Task-level Planning. ArXiv:2511.18203 [cs] version: 4.

Zhang W, Terver B, Zholus A, Chitnis S, Sutaria H, Assran M, Balestriero R, Bar A, Bardes A, LeCun Y and Ballas N (2026) Hierarchical planning with latent world models.

Zhang Y, Xue T, Razmjoo A and Calinon S (2025) Learning Problem Decomposition for Efficient Sequential Multi-Object Manipulation Planning. *IEEE Robotics and Automation Letters (RA-L)* 10(12): 13367–13374.

Zhao Z, Cheng S, Ding Y, Zhou Z, Zhang S, Xu D and Zhao Y (2025) A Survey of Optimization-Based Task and Motion Planning: From Classical to Learning Approaches. *IEEE/ASME Transactions on Mechatronics* 30(4): 2799–2825.

Zhao Z, Lee WS and Hsu D (2023) Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems (NeurIPS)* 36: 31967–31987.

Zhou Z, Garg A, Fox D, Garrett CR and Mandlekar A (2024) SPIRE: Synergistic Planning, Imitation, and Reinforcement Learning for Long-Horizon Manipulation. In: *Conference on Robot Learning (CoRL)*.

Zhu Y, Stone P and Zhu Y (2022) Bottom-Up Skill Discovery From Unsegmented Demonstrations for Long-Horizon Robot Manipulation. *IEEE Robotics and Automation Letters (RA-L)* 7(2): 4126–4133.