Advances in Cognitive Systems XX (2026) 1–16 Submitted 8/2026; published 10/2026

Learning Skills and Affordances for Teleoreactive Control

Pat Langley PATRICK.W.LANGLEY@GMAIL.COM
David Ménager DHMENAGER@GMAIL.COM
Institute for the Study of Learning and Expertise, 2164 Staunton Court, Palo Alto, CA 94306 USA

Abstract

In this paper, we present a novel approach to learning skills and affordances for physical settings. We review PUG/U, a cognitive architecture for embodied agents that stores knowledge about concepts, skills, processes, and motives. The framework uses this content for conceptual inference, continuous control, mental simulation, motion planning, and task planning. After this, we describe PUG/L, a new version of the architecture that encodes skills with qualitative control functions, defines affordance concepts that serve as their conditions, and supports parallel execution of skills and subskills that achieve these conditions. Most important, we report analytic methods for learning new skills and affordance concepts from background knowledge about concepts and processes. We demonstrate these mechanisms on scenarios from a two-dimensional environment in which a robot must approach a target object while avoiding obstacles. In closing, we discuss related research on representing, using, and learning knowledge for teleoreactive control, along with directions for additional research.

1. Introduction

The cognitive architecture movement attempts to formulate unified theories of the mind (Laird, Langley, & Rogers, 2009). There has been substantial progress in the area since its launch in the 1970s, but most frameworks have remained focused on central cognition rather than on interactions between mental processing and an external environment. This is not surprising given that early architectures concentrated on accounts of problem solving and memory. Some efforts have integrated sensory and motor abilities (Mininger & Laird, 2019; Trafton et al., 2013), but these elements seldom play roles as core parts of a theory, despite producing some impressive results with both simulated and physical robots.

This paper focuses on PUG, an architecture that adopts a different perspective. The framework supports classic cognitive structures like beliefs, goals, and intentions, but it also operates in continuous environments that require fine-grained control. To this end, PUG unifies ideas from symbolic task planning and numeric feedback control, We review the architecture's representations and processes, then identify limitations and describe extensions that address them. The latter include introduction of qualitative skills, affordance concepts, parallel execution, and mechanisms for learning new concepts and skills. We demonstrate these extensions on scenarios from a simulated robotics domain. We conclude by discussing related work and directions for future research.

2. Review of the PUG Architecture

PUG is a cognitive architecture, and associated implementation, that has been under development for a decade (Langley et al., 2016, 2017, 2022, 2024). It borrows key ideas from earlier frameworks like ACT-R (Anderson & Lebiere, 1998) and Soar (2012), but especially from ICARUS (Choi & Langley, 2018). Like ICARUS, PUG focuses on agents like mobile robots and self-driving vehicles that operate in with continuous physical environments that contain discrete objects. The two architectures also share a strong commitment to teleoreactive control of agent actions (Nilsson, 1994). In this section, we review PUG's primary theoretical assumptions.

2.1 Representation in PUG

Like other cognitive architectures, PUG distinguishes between stable, long-term memories and dynamic, short-term ones. The framework assumes four varieties of long-term knowledge:

- *Concepts* define categories in terms of relations among entities, as being at an object or facing it. Each concept includes a head with a predicate and arguments, a set of elements with attributes, a set of conditions, and a veracity function.
- *Skills* specify conditional actions for achieving goals, such as moving to an object or turning to face it. Each skill includes a head with a predicate and arguments, a set of elements, a set of conditions, a control function, and a target concept.
- *Processes* describe how control and environmental attributes influence the environment, such as turns changing angles. Each process includes a head with a predicate and arguments, a set of elements, a set of conditions, and a set of difference equations.
- *Motives* specify the conditional utilities for relational concepts, such as the negative value of collisions. Each motive includes a head with a predicate and arguments, a set of elements, a set of conditions, and a utility function.

Table 1 give examples of concepts, skills, processes, and motives for a simulated rover domain from previous PUG papers. These cognitive structures encode complementary types of long-term knowledge that work in unison to support goal-directed but reactive behavior in physical environments.

PUG also includes four distinct types of short-term structures that change during the agent's pursuit of its goals. These include:

- *Beliefs* are instances of concepts. Each belief specifies a predicate with arguments, such as *(robot-at ^id (R1 O1))*, a veracity score (between zero and one), and a positive or negative utility.
- *Intentions* are instances of skills. Each intention contains a predicate with arguments, such as *(move-to R1 O1)*, and a target belief, such as *(robot-at ^id (R1 O1))*.
- *Motion plans* are sets of intentions the agent has committed to execute in parallel, such as *((move-to R1 O1) (turn-right-to R1 O1))*, that determine a trajectory in space over time.
- *Task plans* are sequences of motion plans, organized into AND trees, that specify how to achieve a set of goals (desired beliefs) and their supporting subgoals.

These structures are organized hierarchically, in that task plans incorporate motion plans, which refer to intentions that in turn point to desired beliefs. PUG also generates short-term analogs to processes and motives, but it does not store them in an explicit memory.

Table 1. Examples of four long-term PUG structures for a rover domain, including (a) a concept for being at an object, (b) a skill for moving toward an object, (c) a process that describes changes caused by moving, and (d) a motive that specifies the utility of being at an object.

```
(a) ((robot-at ^id (?r ?o) ^distance ?d)
      :elements   ((robot ^id ?r ^radius ?rr)
                   (object ^id ?o ^distance ?d ^radius ?or))
      :binds      ((?b (- ?d (+ ?rr ?or))))
      :veracity   (cond ((> ?b 10.0) 0.0)
                        ((= ?b 0.0) 1.0)
                        (t (/ (- 10.0 ?b) 10.0))))
(b) ((move-to ?r ?o)
     :elements   ((robot ^id ?r) (object ^id ?o))
     :conditions ((robot-oblique ^id (?r ?o)))
     :control    ((robot ^id ?r ^move-rate (* 0.5 $MISMATCH)))
     :target     ((robot-at ^id (?r ?o))))
(c) ((move-wrt-object ?r ?o)
     :elements ((robot ^id ?r ^move-rate ?m)
                (object ^id ?o ^distance ?d ^angle ?a))
     :changes  ((object ^id ?o ^distance (*dd ?d ?a ?m) ^angle (*da ?d ?a ?m))))
(d) ((robot-at ^id (?r ?o))
      :conditions ((robot ^id ?r ^radius ?rr)
                   (object ^id ?o ^type target ^distance ?d ^radius ?or))
      :function   (cond ((< ?d (+ ?rr ?or 0.25)) 10.0) (t 0.0))
      :type       achievement)
```

2.2 Processing in PUG

Like other cognitive architectures, PUG operates in discrete cycles, each of which compares long-term knowledge to short-term elements. Matched structures add new elements to dynamic memory, which become available for matching on the next round, with cycles continuing indefinitely. This process involves symbolic pattern matching, as in Soar, ACT-R, and ICARUS, but also computes numeric values for veracities, utilities, and control attributes. The latter reflect the architecture's commitment to goal-directed but reactive agents that operate over time in physical environments.

However, PUG differs from other cognitive architectures by organizing its processing into multiple layers, with each level matching long-term structures against elements produced by those below it. These include, from the lowest to highest:

- *Belief processing*, which compares concepts to percepts from the environment and existing beliefs to infer new beliefs, each with an associated veracity. Each cycle at this level generates a single belief, which may support others on later cycles.
- *State processing*, which compares active intentions and processes to elements in the current belief state. This calculates control values for each active intention, sums values for each control attribute, and makes predictions about the next environmental state.

- *Mental simulation*, which invokes state processing repeatedly to generate a sequence of states that describes a trajectory over time. On each cycle, it also matches motives to current beliefs to compute utilities, which it sums for associated with each state.
- *Motion planning*, which starts with a single intention and carries out greedy search through the space of intention sets. On each cycle, it uses mental simulation to generate candidate trajectories and guides its selection based on the their utilities.
- *Task planning*, which starts from goal (desired beliefs with low veracities) and uses means-ends analysis to construct a plan to achieve them. On each cycle, it retrieves relevant skills and generates a motion plan for each, again guiding search with utilities.

Thus, PUG is a *cascaded* architecture in which each level of processing builds on those below it by accessing elements in a shared dynamic memory. Together, these layers support teleoreactive behavior in physical environments. Recent papers (Langley et al., 2022, 2024) reports tests of the framework in a simulated robotics domain.

3. The Extended PUG Architecture

Research on PUG has steadily added new capabilities while retaining its core commitment to utility-guided teleoreactive control in continuous domains (Langley et al., 2016, 2024). However, the current version still has limits that we should address. These revolve around the fragility of skills, which have depended on careful specification of control functions, all-or-none conditions on skill application, and the absence of acquisition mechanisms. In this section, we address the first two issues in ways that lay the groundwork for skill learning, which we will discuss later.

3.1 Qualitative Skills

The first major revision concerns skill representation. Research on PUG has aimed to unify ideas from symbolic planning and continuous control, so skills incorporate control functions that determine the value of control attributes on each decision cycle. However, these have proven difficult to craft in ways that produce robust behavior. The revised architecture instead assumes that:

- Skills specify *qualitative* control functions that indicate only the signs for control values.
- If a skill has a control function for attribute C with sign S and maximum value M, and if its target concept has veracity V, then the value for C is $S \times M \times (1 - V)$.

For example, consider the modified skill for moving toward a target object that appears in Table 2 (a), which differs from its analog in Table 1. The *:control* field contains the expression *(robot ^id ?r ^move-rate +)*, which states only that *move-rate* should have a positive value (for moving forward), but PUG must still compute a continuous value for this control attribute.

The architecture's response is to multiply the *+* sign by the known maximum value of the control attribute and by the current error signal, which is one minus the veracity of the target concept. Thus, if the maximum *move-rate* is 2.0 and the veracity for the target belief *(robot-at ^id (r1 o1))* on the current decision cycle is 0.8, then PUG would compute $1.0 \times 2.0 \times 0.8 = 0.4$ as the *move-rate*. In this way, the extended framework still supports continuous control despite adopting qualitative encodings, which we will see later simplifies skill learning.

Table 2. Examples of structures in the extended PUG architecture for the rover domain. These include (a) two qualitative skills for moving to an object and turning toward it, as well as (b) two affordance concepts that appear as the conditions for their respective skills.

```
(a) ((move-to ?r ?o)
     :elements     ((robot ^id ?r) (object ^id ?o))
     :conditions   ((affords-move-to ^id (?r ?o)))
     :control      ((robot ^id ?r ^move-rate +))
     :target       (robot-at ^id (?r ?o)))

    ((turn-right-to ?r ?o)
     :elements     ((robot ^id ?r) (object ^id ?o))
     :conditions   ((robot-left-of ^id (?r ?o)))
     :control      ((robot ^id ?r ^turn-rate +)
     :target       (affords-move-to ^id (?r ?o)))
```

```
(b) ((affords-move-to ^id (?r ?o))
     :elements     ((robot ^id ?r ^move-rate ?m)
                    (object ^id ?o ^distance ?d ^angle ?a))
     :binds        ((?u (+ (*dd ?d ?a ?m))))
     :veracity     (cond ((< ?u 0.0) 0)
                         ((= ?u ?m) 1.0)
                         (t (/ (- ?m ?u) ?m))))

    ((affords-turn-right-to ^id (?r ?o))
     :elements     ((robot ^id ?r ^turn-rate ?t) (object ^id ?o))
     :binds        ((?u (+ (* -1.0 ?t))))
     :veracity     (cond ((< ?u 0.0) 0)
                         ((= ?u ?t) 1.0)
                         (t (/ (- ?u ?t) ?t))))
```

3.2 Affordance Concepts

Another drawback was that PUG only supported all-or-none conditions on skill application, such as numeric thresholds on state attributes like distance or angle to an object. The syntax for skills allowed conditions on relational concepts, but this was seldom used because it was unclear how to interpret conditions that could match to varying degrees. However, recall that a PUG agent executes a skill in order to achieve its associated target concept and that this should happen only when such action will increase the latter's veracity. Unless external factors intervene, we want a skill's execution to cause monotonic progress toward this end.

This suggested that we incorporate the psychological notion of *affordance*. Gibson (1977) introduced this term to refer to how well an object enables certain actions. For instance, a chair affords the activity of sitting, while a hammer affords driving a nail into wood. Langley et al. (2018) have argued that affordances are *graded*, in that an objects enable actions to greater or lesser degrees. This idea maps well onto PUG's notion of veracity, which ranges from zero to one. In response, we introduced two additional assumptions about skills and concepts to the architecture:

- Skills have *affordance conditions*, which refer to concepts that match to differing degrees.
- If the veracity of a skill's affordance condition is nonzero, then executing this skill in isolation will increase the veracity of its target concept.

For example, consider the condition *(affords-move-to ?r ?o)* in the skill from Table 2 (a). The table also shows the definition for this concept, which refers to **dd*, a trigonometric function that takes distance *?d*, angle *?a*, and current move-rate *?m* as arguments.

The concept includes a conditional statement for computing its veracity in particular situations. When *(*dd ?d ?a ?m)* is zero or less, the veracity is zero, which means that applying *(move-to R1 O1)* will not bring the robot closer to the object. When the expression equals the maximum value for *move-rate*, then the robot will approach the object as fast as possible. And when *(*dd ?d ?a ?m)* falls between zero and one, the robot will approach the object at a rate proportional to this value. Thus, including this affordance condition in the *move-to* skill will ensure the desired monotonic improvement, provided, that is, the trigonometric function **dd* is defined appropriately.

3.3 Parallel Skill Execution

Given that an affordance condition A on some skill specifies when executing it will produce progress toward its target concept, this raises the question of how to proceed when A's veracity is less than one. The revised architecture adopts two principles for this situation:

- When the affordance condition C for skill S1 has veracity zero, the agent does not execute S1 but instead applies another skill S2 that would achieve C.
- When the affordance condition C for skill S1 has veracity between zero and one, the agent executes both S1 and a skill S2 that would increase C's veracity.

These assumptions mean not only that two or more skills can apply at the same time, which PUG already supports in its motion plans, but that they can execute in parallel specifically when one skill has been invoked in order to satisfy a condition of the other skill.

For example, suppose the agent has the goal *(robot-at ^id (r1 o1))* but the robot is 3.0 units away from O1 and at a 120 degree angle from it. In this situation, the belief *(affords-move-to ^id (R1 O1))* has a veracity of zero, so that executing *(move-to R1 O1)* will not bring the robot closer to its target. In response, PUG would initiate another intention, *(turn-right-to R1 O1)*, that increases the veracity of the affordance condition. When the score exceeds zero, PUG will execute *(move-to R1 O1)*, moving the robot forward, but also keep applying *(turn-right-to R1 O1)*. This will continue until the robot directly faces the target object, at which point it stops turning and only moves forward.

Classic systems for task planning assume that, when carrying out action A1 would produce a state that satisfies the conditions of another action A2, the agent must complete its execution of A1 before it begins execution of A2. This holds even for planning systems that support durative skills that take multiple time steps to complete. However, they adopt this contraint mainly because they assume the relations that describe a state are either true or false, rather than being true to differing degrees. PUG's assignment of veracities to beliefs is what enables the architecture to execute in parallel skills that nevertheless involve such logical dependencies.

4. Learning New Skills and Concepts

As noted above, previous versions of PUG have taken strong positions about how to represent knowledge and how to use that content to support teleoreactive behavior in physical settings. However, they have lacked a claim that is central to many other cognitive architectures: how to acquire this knowledge. In this section, we present some general constraints on learning, after which we describe methods for acquiring two types of knowledge elements.

4.1 Theoretical Constraints on Learning

The extended architecture, PUG/L, incorporates mechanisms for acquiring two types of mental structures: skill and concepts. However, before describing their details, we should state some constraints on their operation that differ from most research on machine learning. These include four generic postulates about human-like behavior:

- Expertise is acquired in a *piecemeal manner*, with one element being added at a time.
- Learning is an *incremental activity* that processes one experience at a time.
- Learning is *guided by knowledge* that aids the interpretation of new experiences.
- Cognitive structures are acquired and refined *rapidly*, from small numbers of training cases.

We have taken these from Langley (2022), who argued that the first three capabilities support the final one. These assumptions are rare in recent AI research, but they held for early work on machine learning and they remain common within the cognitive systems paradigm.

The PUG/L architecture incorporates these postulates but also makes two further theoretical commitments that are linked to its support for teleoreactive control in physical environments:

- Learning is *problem driven*, in that it occurs when lack of relevant knowledge keeps the agent from achieving one of its goals.
- Learning is *analytical*, in that it relies on the compilation of existing knowledge into a more usable form that enables goal achievement.

These two premises are even less common in modern machine learning, but they are shared by many cognitive architectures, including Soar (Laird, 2012) and ACT-R (Anderson & Lebiere, 1998). However, we will see that they take quite different forms that are driven by PUG's other theoretical commitments.[1] We should clarify that the architecture, as it stands, does not address acquisition of processes or motives, which it treats as given. Nevertheless, we have discussed methods for learning them elsewhere (Langley, 2019, 2024) and we hope to incorporate them in future versions.

4.2 Acquiring Qualitative Skills

Skill acquisition in PUG/L occurs in the context of task and motion planning. Briefly, it takes place when the agent wants to achieve some goal but lacks a skill that would let it do so. In response, the architecture takes advantage of existing knowledge to construct a new cognitive structure that should let it reach this end. We can state the problem as:

1. We will not claim that all skill acquisition occurs in this manner; empirical learning from observation and experimentation are complementary mechanisms on which there has been considerable research (e.g., Li et al., 2009).

- Given: A set of conceptual rules with associated veracity equations
- Given: A set of processes with effects of state and control attributes
- Given: A goal G to achieve an instance of some known concept C
- Find: A skill that, when executed, will make progress toward goal G

This new skill must specify a target concept based on the goal, a set of relevant state and control attributes, a qualitative equation for a relevant control attribute, and an affordance condition under which the skill will make progress toward the goal.

To construct a new skill S from a goal G, known processes, and known concepts, the extended architecture first carries out six preliminary steps. These include:

- Creating an initial version of S with empty fields;
- Putting a generalized version of goal G in S's :target field;
- Accessing the :veracity equation for S's target concept T;
- Extracting all attributes {A} that determine T's veracity;
- Retrieving all processes {P} that influence attributes in {A}; and
- Extracting all state and control attributes {B} that occur in {P}.

Once the system has collected these pieces of information, it takes four additional steps to populate the new skill's various fields by:

- Placing the attributes in {B} in the skill S's *:elements* field;
- Extracting all control attributes {C} that occur in {P};
- Including a qualitative control equation for each control attribute in {C}; and
- Inserting a new affordance concept N into S's *:conditions* field

Together, these actions produce a qualitative skill that, upon execution in the environment or during mental simulation, will make progress toward the goal that motivated its creation.

We can clarify this learning mechanism with an example that involves learning a skill for achieving the goal *(robot-at ^id (R1 O1))*. Assume that the agent begins with knowledge about the process *move-wrt-object* from Table 1, which describes the effects of moving in the forward direction. This assumes an egocentric representation that encodes objects in terms of their distance and angle from the robot. The trigonometric functions **dd* and **da* specify how to compute changes in an object's distance and angle when the robot moves forward. Further suppose the agent begins with a definition for the concept *robot-at* from Table 1. This specifies two elements, the robot and an object, and a veracity expression that refers their distance:

```
(cond ((> ?d 10.0) 0.0) ((= ?d 0.0) 1.0) (t (/ (- 10.0 ?d) 10.0))).
```

This means that when distance ?d is 0.0, the veracity is 1.0, when ?d is 5.0, the veracity is 0.5, and when ?d is 10.0, the veracity is 0.0. In other words, the degree to which *robot-at* holds is a piecewise linear function of the robot's distance from the object.

Now suppose the agent has the goal *(robot-at ^id (R1 O1))* but lacks any skills that would achieve it. In response, PUG/L would create a new skill with the same arguments as the goal: *(move-to ?r ?o)*. After this, it would insert an *:elements* field with descriptions for these arguments, along with their associated attributes: *((robot ^id ?r ^move-rate ?m) (object ^id ?o ^distance ?d ^angle ?a))*. Next it would specify the skill's *:control* field, which includes a sign for each control

attribute that appears in the processes that affect the veracity of the skill's target concept. In this case, the resulting control expression is *((robot ^id ?r ^move-rate +))*. Finally, the system adds a condition that determines when the skill is applicable. This has a new predicate and takes the same arguments as the skill, such as *(affords-move-to ^id (?r ?o)*. The result is the first skill in Table 2 (a), which we discussed in Section 3.1.

4.3 Learning Affordance Concepts

However, the new skill is not operational without a definition for its affordance concept to determine when it should apply. PUG/L constructs this in a similar way by taking advantage of background knowledge about processes and other concepts. We can state the problem as:

- Given: An initial set of conceptual rules with veracity equations
- Given: A set of processes with effects of state and control attributes
- Given: A new skill S for achieving a target concept C
- Find: A concept that serves as the affordance condition for S

Recall that an affordance condition specifies a class of situations in which a skill, if executed, will increase the veracity of its target concept. That is, it will match with nonzero veracity only when the control equation will improve the match of the target concept.

This new concept must specify (a) a set of relevant state and control attributes and (b) a veracity function specified in terms of these attributes. To define the affordance concept C for a new skill S, the architecture carries out six further steps. These include:

- Extracting algebraic expressions in {P} that affect attributes in {A};
- Substituting their sum into T's veracity function to produce E;
- Making expression E the core :veracity function V for concept N;
- Specifying that veracity V has the value zero when this sum is zero or less;
- Indicating that veracity V has the value one when this sum is maximized; and
- Specifying how to interpolate veracities between zero and one.

The result of these steps will be the definition of a new affordance concept, named in the recently learned skill, that states when executing this skill will have the desired effect.

We can illustrate this mechanism by continuing the earlier example, which involves learning a skill for achieving (robot-at ?r ?o). As before, PUG/L generates the head in terms of the new predicate and its arguments: *(affords-move-to ^id (?r ?o))*. Next the architecture populates the *:elements* field for the new concept, which is the same as that for its associated skill. To compute the concept's veracity function, PUG takes the sum of effects from all processes that influence the associated skill's target concept. In this case, only the term *(*dd ?d ?a ?m)* from *move-wrt-object* is relevant and inserted into the *:binds* field. The system then inserts a function that uses a bound variable, say *?u*, to compute veracity:

```
(cond ((< ?u 0.0) 0) ((= ?u ?m) 1.0) (t (/ (- ?m ?u) ?m))))
```

This function has three conditions. The first states when the concept's veracity is zero, the second when it is one, and the third when it falls between these two extremes. Since the concept's purpose is to tell when the skill will be effective, the maximum veracity occurs when the control attribute *move-rate* will have full effect. The final result is the first affordance concept in Table 2 (b).

4.4 Recursive Skill and Concept Acquisition

Once PUG/L has defined a new skill and its associated affordance concept, the task planner will attempt to use them to elaborate a plan that achieves its original goal. Since this relies on means-ends analysis, it will create a subgoal to achieve an instance of the affordance concept. However, in some cases the agent may lack a relevant skill to achieve this goal as well, which will lead in turn to a recursive call on the learning module. The same type of retrieval failure can occur during motion planning, which instead carries out greedy search for a set of intentions that achieve the primary goal while retaining high utility along the way.

This situation arises in our earlier example. The task planner uses the new skill, *move-to*, to create the intention *(move-to R1 O1)*, which has the condition *(affords-move-to r1 o1)*. But the system lacks any skill that would let the agent achieve this goal, so it again uses knowledge about a different process, *turn-wrt-object*, to create another skill, *turn-right-to*, which appears as the second structure in Table 2 (a). This in turn leads the system to acquire another concept that specifies when *turn-right-to* will increase the veracity of its target concept *affords-move-to*. This step produces the second affordance concept in Table 2 (b). This skill and concept are appropriate when the robot is facing to the left of the target object. When it is facing to the right, then it would learn an analogous skill with $-$ as the sign in the control equation and with 1.0 as coefficient in the concept.

4.5 Summary

In summary, the extended architecture includes mechanisms for learning new skills and associated affordance concepts from background knowledge. PUG/L invokes these techniques when it attempts to achieve a goal for which it lacks skills, which can occur during either task planning or motion planning. Rather than relying on induction from training cases, the system compiles content from relevant concepts and processes into the cognitive structures it creates.

When the architecture also lacks a skill for achieving a subgoal involving an affordance concept, it calls the learning mechanism recursively, continuing until it reaches a subgoal it can handle with existing skills. Another complication, not discussed above, occurs when multiple processes influence the veracity of a learned skill's target concept. In such cases, the veracity function for its associated affordance concept will include a sum of terms from those processes.

These two complementary mechanisms – acquiring qualitative skills and affordance concepts – support piecemeal, incremental learning that is guided by background knowledge, which in turn leads to rapid improvement in agent performance. Moreover, they build on other extensions like qualitative control functions and parallel skill execution. Together, they enable human-like learning of the rich cognitive structures needed for teleoreactive behavior in physical environments.

5. Demonstrations of the Extended Architecture

To demonstrate the extended architecture's functionality, we ran a PUG/L agent on three navigation scenarios similar to those reported by Langley and Katz (2024). The required a simulated robot to approach a visible target, sometimes altering its heading and avoiding obstacles. The primary differences from earlier studies concerned use of qualitative skills with affordance concepts, as well

as acquisition of both types of structures. Each reported run begins with no skills and a limited set of concepts, so the system must these structures before it can use them to achieve its goals.

5.1 Domain Specification

The demonstrations used a continuous 10×10 two-dimensional world. A single robot controlled by the agent had a radius of 0.15 units, a maximum move rate 0.1, a maximum turn rate 5.0, and an initial fuel level of 50. Each target was a stationary circle of radius 0.4. A perceptual simulator returned, on each execution cycle, a set of objects described by their attributes, including their distance and angle from the robot in polar coordinates. The concept *robot-at* reached veracity one when the distance between centers was no greater than the sum of their radii and this quantity increased as the robot approached surface contact.

The processes relevant to all three scenarios were *move-wrt-object*, which predicts changes $d' = d + \Delta_d(d, a, m)$ and $a' = a + \Delta_a(d, a, m)$ from distance d, angle a, and move rate m, and *turn-wrt-object*, which predicts $a' = a - t$ from turn rate t. For the obstacle scenario, we also provided the agent with the concepts *in-collision-cone*, *robot-colliding*, and *robot-penetrating*. Motives assigned negative utility to the last two relations, which led motion planning to introduce maintenance subgoals when the robot would hit an obstacle.

5.2 Learning to Approach an Object Directly

In the first scenario, the robot begins facing object *O1*, at a distance of 3.0 units and angle 0.0, with the goal *(robot-at ^id (R1 O1))*. However, the agent lacks any skill that would achieve the *robot-at* relation, so it invokes its learning mechanism to construct one, along with an affordance concept that specifies when executing it will produce progress toward this goal.

The only challenge in learning the skill is identifying the relevant control attribute (*move-rate*) and determining its sign (*+*), which it obtains by analyzing the definition of *robot-at*. The result is *move-to*, the first skill shown in Table 2. To create an affordance concept, PUG retrieves the only process, *move-wrt-object*, that affects distance, generates a veracity function, and generates *affords-move-to*, the first concept in Table 2.

Once PUG/L has added these structures, the execution module recognizes that *(affords-move-to R1 O1)* has veracity 1.0, which means that *(move-to R1 O1)* is immediately applicable. This situation continues to hold on future execution cycles, with the robot's movement toward the target object reducing its distance and increasing the veracity of *(robot-at ^id (R1 O1))*. The robot goes on in this vein until it finally reaches object *O1*, at which point its movement halts, having followed the trajectory in Figure 1 (a).

5.3 Learning to Turn and Approach

In the second scenario, the agent again has goal *(robot-at ^id (R1 O1))*, but it begins at distance of 3.0 units and an angle of 90° with respect to O1. As before, it has no skill for *robot-at* and failure to retrieve one lead it to invoke learning. The same analysis produces the first skill in Table 2, *move-to*, and the first affordance concept, *affords-move-to*.

However, this time the affordance condition is not satisfied, having a veracity of zero, which indicates that executing *move-to* will not have the desired effect. Thus, PUG attempts to retrieve a skill that would satisfy *(affords-move-to ^id (R1 O1))*, but none exists, leading it to construct a new skill that would achieve this relation and an associated affordance concept. As before, creating the skill is not difficult, but the system must identify the control attribute (*turn*) and its sign (*+*), giving the second skill in Table 2, *turn-right-to*. PUG/L also creates the second concept in Table 2, *affords-turn-right-to*, using the process *turn-wrt-object*, which affects an object's angle.

Once it has created all four structures, the execution module realizes that the affordance condition for *(turn-right-to R1 O1)* is satisfied but the one for *(move-to R1 O1)* is not, so it executes the former but not the latter. This continues until the veracity of *affords-move-to* exceeds zero, at which point it applies both intentions in parallel. Eventually, its veracity approaches 1.0, so the agent stops turning but continues to move forward until reaching the target object. Figure 1 (b) depicts the trajectory that the robot follows during this run.

5.4 Learning to Avoid Obstacles

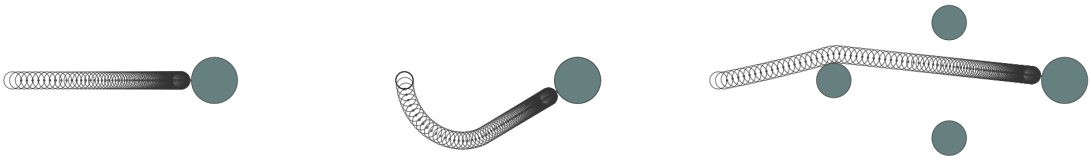
In the third scenario, the agent has a similar goal, *(robot-at ^id (R1 O4))*, and starts facing the target object, but there is an obtacle, *O1*, immediately ahead and two others, *O2* and *O3*, to the left and right. As in the first run, the agent attempts to retrieve a relevant skill without success, so it uses the mechanism already discussed to create one that will achieve the goal, along with an associated affordance concept. This condition is already satisfied, so the agent begins to execute the newly acquired skill immediately.

Unfortunately, mental simulation reveals that using this skill alone would leave *O1* in the robot's collision cone, which it realizes using the concept *in-collision-cone* provided as background knowledge. This trajectory has negative utility, so the motion planner attempts to retrieve a skill that will avoid it, but none is available. To remedy this situation, the agent learns a second skill with a target concept based on *(not (in-collision-cone ^id (R1 O1)))*. This includes the control attribute *turn-rate* and sign *+*, which will move it out of the collision code, and an appropriate affordance condition.

The agent notes that it can execute the new skill right away, as its affordance concept has nonzero veracity, but it also notes that another, supporting, skill is possible. The result is a third skill, created recursively, with control attribute *move-rate* and sign *+*, that will let the robot move out of the collision cone more rapidly. The affordance condition for this skill is also satisfied, so PUG executes them in parallel until they take the robot out of the collision cone. However, swerving around *O1* leads it into the cone of *O2*, which leads to another skill with control attribute *move-rate* but sign *+*, which takes it in the other direction. Eventually, the agent escapes from both cones and heads to *O4*, the target object. Figure 1 (c) shows the resulting trajectory.

5.5 Summary

Together, the demonstrations show three depths of processing for the acquisition of skills and affordance concepts, with each scenario subsuming the previous one. In the first run, PUG/L learns a single skill-affordance pair, which suffices to achieve its goal. The second scenario illustrates both the recursive construction of skills with affordance concepts and their parallel execution. The most



(a) Move straight ahead **(b)** Turn and move ahead **(c)** Avoid obstacles

Figure 1. Trajectories generated with learned concepts and skills in the three scenarios: (a) The robot moves directly to the target using one learned skill; (b) The robot turns before moving to the target using two learned skills linked by an affordance concept; and (c) The robot moves to the target while avoiding collisions using a move skill, an obstacle avoidance skill, and associated affordance concepts. The large filled circle is the target, smaller filled circles are obstacles, and the clear circles show the robot's trajectory.

complex case involves acquisition and use of competing skills, two of them with negated target concepts, that the architecture introduces to avoid negative-utility trajectories.

In each run, system logs record why skill retrieval failed, which goal invoked learning, and when the created skills became applicable because their affordance condition was satisfied and when they halted on achieve their goal. These results provide evidence that a single analytical acquisition mechanism, invoked during either task planning or motion planning, can handle a variety of physical situations. Moreover, learning occurs in an incremental and piecemeal that draws on background knowledge, much as in humans.

6. Related Research

The extended PUG architecture combines ideas from a number of earlier traditions. For example, there has been work on qualitative approaches to control, include Nilsson's (1994) efforts on teleoreactive programs. There have even been some systems that learn qualitative controllers, from observation or experimentation, by inducing decision trees (Šuc et al., 2004) or logical control rules (Benson, 1995). However, these invoke discrete actions in continuous environments, whereas PUG/L adopts a qualitative representation that combines signs, error signals, and maximum values to calculate continuous control values.

There has also been research on representing, recognizing, and learning affordances in physical settings (e.g., Allevato et al., 2018; Mo et al., 2022). However, this work has typically focused on supervised learning from visual data or feedback from extensive simulations to induce predictive models. PUG's notion of affordance is linked instead to whether execution of durative skills makes progress toward goals, and acquisition relies on reasoning over background knowledge about when this will occur rather than large training sets.

This analytical approach to learning bears marked similarities to those in some cognitive architectures. Soar creates new chunks in response to impasses by analyzing reasoning chains, but it does not explicitly compile knowledge elements in the same way as PUG/L. ACT-R uses a knowledge compilation mechanism that creates new production rules by composing existing ones, but it is not

problem driven and it produces qualitative structures that engage in direct actions, rather than ones that compute numeric values. The same holds for ICARUS, which creates qualitative skills that build on ones already in long-term memory.

Two other related efforts bear discussion. Brown and Sammut (2011) report a system that learns to use tools for achieving, drawing on knowledge of existing skills to identify new skills in an expert's behavior. However, their approach relies on abduction rather than compilation to create structures and runs experiments to refine them. DeJong and Oblinger's (1993) work comes closer, it uses known processes to explain observed behavior and it learns to generate multivariate trajectories. Their system combines ideas from symbolic reasoning and continuous control, but also uses curve fitting to estimate parameters, rather than creating skills in an analytical, goal-driven manner.

7. Concluding Remarks

In this paper, we presented extensions to PUG, a cognitive architecture for embodied agents that supports teleoreactive behavior. Previous versions have unified continuous control, mental simulation, motion planning, and task planning, but required careful entry of concepts and skills, and offered no account of their acquisition. In response, we introduced PUG/L, which adopts a qualitative representation for skills, conditions their execution on affordance concepts, and supports a new kind of parallel execution. The extended architecture also learns skills and affordance concepts by composing elements of known concepts and processes.

We reported demonstrations of the system's functionality on three scenarios from a two-dimensional rover domain. One of these showed PUG acquiring a qualitative skill for moving the robot to a target object, along with an associated affordance concept. Another run involved learning not only these structures, but also a skill for turning toward an object and its affordance condition. Finally, we showed PUG/L's behavior in a more complex situation that required the robot to maneuver around two obstacles, leading it to create six additional skills and affordance concepts. These runs provided encouraging evidence that the extensions operate as intended.

Nevertheless, our results remain preliminary and future efforts should test PUG/L on more complex scenarios with different initial configurations to demonstrate that both the performance and learning extensions are robust. In particular, we should show that acquisition works as planned when multiple processes come into play. We should also address limits of the current implementation. For instance, PUG/L should learn from cases in which goal concepts are disjunctive and it should allow deeper lookahead when evaluating affordances. In the longer term, we should provide it with mechanisms for inducing processes from and update motives based on experience. Together, these will make the architecture a more complete theory of embodied intelligence.

Acknowledgements

The research reported here was supported by Grant No. N00014-24-1-2737 from the US Office of Naval Research and Grant No. FA9550-23-1-0580 from the US Air Force Office of Scientific Research, which are not responsible for its contents.

References

Allevato, A., Thomaz, A., & Pryor, M. (2018). Affordance discovery using simulated exploration. *Proceedings of the Seventeenth International Conference on Autonomous Agents and MultiAgent Systems* (pp. 2174–2176). Stockholm, Sweden.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum.

Benson, S. (1995). Inductive learning of reactive action models. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 47–54). Tahoe City, CA: Morgan Kaufmann.

Brown, S. A., & Sammut, C. A. (2011). Tool use learning in robots. *Proceedings of the AAAI Fall Symposium on Advances in Cognitive Systems* (pp. 58–65). Arlington, VA: AAAI Press.

Choi, D., & Langley, P. (2018). Evolution of the ICARUS cognitive architecture. *Cognitive Systems Research*, *48*, 25–38.

Dejong, G., & Oblinger, D. (1993). A first theory of plausible inference and its use in continuous domain planning. In S. Minton (Ed.), *Machine learning methods for planning*. San Mateo, CA: Morgan Kaufmann.

Gibson, J. J. 1977. The theory of affordances. In R. E. Shaw and J. Bransford (Eds.), *Perceiving, acting, and knowing*. Hillsdale, NJ: Lawrence Erlbaum.

Laird, J. E. (2012). *The Soar cognitive architecture*. Cambridge, MA: MIT Press.

Langley, P. (2019). Scientific discovery, causal explanation, and process model induction. *Mind & Society*, *18*, 43–56.

Langley, P. (2024). From explainable to justified agency. In S. Tulli & D. W. Aha (Eds.), *Explainable agency in artificial intelligence*. Boca Raton, FL: Chapman & Hall/CRC.

Langley, P., Barley, M., Meadows, B., Choi, D., & Katz, E. P. (2016). Goals, utilities, and mental simulation in continuous planning. *Proceedings of the Fourth Annual Conference on Cognitive Systems*. Evanston, IL.

Langley, P., Choi, D., Barley, M., Meadows, B., & Katz, E. P. (2017). Generating, executing, and monitoring plans with goal-based utilities in continuous domains. *Proceedings of the Fifth Annual Conference on Cognitive Systems*. Troy, NY.

Langley, P., & Katz, E. P. (2022). Motion planning and continuous control in a unified cognitive architecture. *Proceedings of the Tenth Annual Conference on Advances in Cognitive Systems*. Arlington, VA.

Langley, P., & Katz, . P. (2024). A unified cognitive architecture for embodied intelligent agents. *Proceedings of the Eleventh Annual Conference on Advances in Cognitive Systems*. Palermo, Italy.

Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, *10*, 141–160.

Langley, P., Sridharan, M., & Meadows, B. (2018). Represention, use, and acquisition of affordances in cognitive systems. *Proceedings of the AAAI Spring Symposium on Integrating Representation, Reasoning, Learning, and Execution for Goal Directed Autonomy*. Stanford, CA: AAAI Press.

Li, N., Stracuzzi, D. J., Langley, P., & Nejati, N. (2009). Learning hierarchical skills from problem solutions using means-ends analysis. *Proceedings of the Thirty-First Annual Meeting of the Cognitive Science Society*. Amsterdam.

Mininger, A., & Laird, J. E. (2019). Using domain knowledge to correct anchoring errors in a cognitive architecture. *Advances in Cognitive Systems*, *8*, 133–148.

Mo, K., Qin, Y., Xiang, F., Su, H., & Guibas, L. (2022). O2O-Afford: Annotation-free large-scale object-object affordance learning. *Proceedings of Machine Learning Research*, *164*, 1666–1677.

Nilsson, N. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, *1*, 139–158.

Šuc, D., Bratko, I., & Sammut, C. (2004). Learning to fly simple and robust. *Proceedings of the Fifteenth European Conference on Machine Learning* (pp. 407–418). Pisa, Italy: Springer.

Trafton, J. G., Hiatt, L. M., Harrison, A. M., Tamborello, F., Khemlani, S. S., & Schultz, A. C. (2013). ACT-R/E: An embodied cognitive architecture for human robot interaction. *Journal of Human-Robot Interaction*, *2*, 30–55.