Proceedings of the Tenth Annual Conference on Advances in Cognitive Systems Submitted 9/2022; published 11/2022

Motion Planning and Continuous Control in a Unified Cognitive Architecture

Pat Langley LANGLEY@STANFORD.EDU
Center for Design Research, Stanford University, Stanford, CA 94305 USA

Edward P. Katz EDPKATZ@STANFORD.EDU
Stanford Intelligent Systems Laboratory, Stanford University, Stanford, CA 94305 USA

Abstract

In this paper, we present a theory that unifies ideas from symbolic reasoning and continuous control. The approach builds on the earlier PUG architecture, but also makes new contributions: symbolic concepts can match to different degrees; symbolic skills include control equations that are modulated by mismatches of target concepts; skills can operate in parallel and processes can predict future states to support mental simulation; and the resulting trajectories, annotated with utilities, guide search for effective motion plans. We also describe PUG/C, an implemented version of this theory and demonstrate its behavior in simulated robot environments. Finally, we review connections to prior work that bridges the two paradigms and propose directions for further extensions.

1. Introduction and Motivation

Early research in artificial intelligence focused on high-level tasks such as problem solving and multi-step reasoning, and the subfield of cognitive architectures (Langley, Laird, & Rogers, 2009) has continued this tradition. However, humans also operate in physical environments that require them to exert fine-grained control over their effectors to achieve concrete goals. In addition, early efforts emphasized the role of symbolic mental structures, including use of domain knowledge to guide behavior, which the cognitive architecture movement has also adopted as a core principle. Yet physical environments have a continuous character that is difficult to handle with discrete representations alone. This suggests that we should extend architectural designs to remedy this oversight.

In contrast, the cybernetics and control communities (Bennett, 1996) have directly addressed fine-grained application of effectors in the external world, supporting this ability with continuous state encodings and quantitative control equations. But the control framework on its own offers no obvious pathway to high-level cognitive processes, including the problem solving that is needed to achieve many physical goals, and it ignores the clear benefits of using relational representations to describe situations and expertise. Moreover, the paradigm has focused its energies on low-level behavior, rather than attempting to develop unified accounts that describe the role of continuous processing in more complete intelligent agents.

There has been a substantial body of work in robotics on combining task and motion planning (Garrett et al., 2021) that links the two levels of description. These join continuous techniques for

guiding robot movements with discrete ones for generating action sequences to support complex robot activities. However, such efforts have concentrated on their integration rather than on unification, which is the objective of research on cognitive architectures. In this paper, we present a deeper account that merges ideas from the discrete, symbolic paradigm and the continuous, numeric framework. Our research objective is to advance *theories* of embodied intelligent agents rather than to develop methods that outperform existing ones in experimental studies.

We start by describing the target behaviors that we aim to replicate and reviewing classic approaches to producing them. Next we review PUG, a cognitive architecture that took some steps toward unification but that also had important limitations. In response, we then present a revised theory that addresses the shortcomings and we report PUG/C, an augmented architecture that uses the new assumptions to support fluid reactive control, flexible mental simulation, and continuous motion planning. After this, we report empirical demonstrations of these abilities in a simulated robotics environment. Finally, we reiterate our framework's key ideas and examine their connections to previous research on symbolic AI and continuous control. This includes work in robotics on task planning, although we are concerned here with motion planning and continuous control.

2. Target Behaviors and Theoretical Challenges

Existing cognitive architectures already support task-level planning with qualitative models of actions' effects, typically using heuristic search to find sequences of steps that achieve goals. For example, they can generate a plan that places an object *O* at location *L* by moving a robot to *O*, grasping *O* with its gripper, moving the robot to *L*, and dropping *O* there. However, their reliance on discrete, symbolic representations does not directly support fine-grained motion in continuous settings. We want to extend architectural theory to include abilities like:

- Calculation of the values for quantitative control attributes for particular physical situations and agent goals, such as speed and direction of movement;
- Mental simulation of smooth trajectories in continuous space, such as maneuvering around obstacles that occur on the path to a target object; and
- Heuristic search for motion plans that produce these trajectories, such as deciding to avoid one obstacle on its left and another obstacle on its right.

Although these involve imagined physical behaviors, they are cognitive tasks that require mental representations and processing. And yet they are distinct from task-level planning, which is more naturally described in qualitative terms. In addition, we desire an account that is generally consistent with other findings about human behavior and theories of the cognitive architecture.

Consider the scenario in Figure 1 (a), in which the robot on the left must reach the target object on the right while avoiding collisions with two obstacles between them. The robot senses its distance and angle to each object as it traverses the environment by altering values for control attributes like speed and direction. However, the agent must not simply execute the path shown, but also select it from a space of possible trajectories. The abilities listed above – setting values for control attributes, simulating a candidate path mentally, and considering alternative trajectories – all have roles to play in this decision-making process. The AI and robotics communities have explored two broad paradigms for motion planning – heuristic search and feedback control – that we will review in turn.

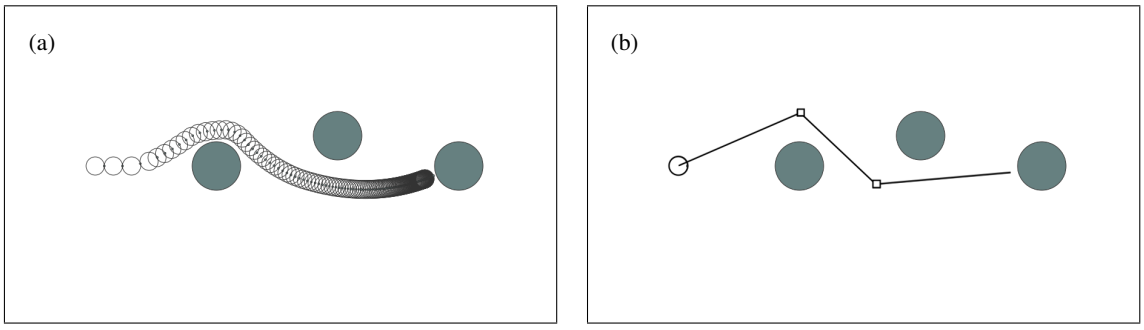

Figure 1. (a) An efficient and natural trajectory for moving the robot on the left to the target object on the right even when no waypoints are provided; (b) an inefficient trajectory that the original PUG architecture would generate for the same task when waypoints are given. In the first plan, turning and moving forward take place in parallel; in the second, they must happen sequentially.

The theory of heuristic search emerged from studies of human problem solving (Selz, 1922; Newell, Shaw, & Simon, 1960). Classic work focused on abstract tasks like puzzles and chess, which one can easily encode in discrete terms, with states encoded as collections of relations and actions as qualitative changes that produce new states. Unfortunately, attempts to adapt this framework to motion and path planning have often produced techniques that bear little resemblance to human behavior. One approach involves discretizing space into a fine-mesh grid, which would produce many cells for the task in Figure 1 (a). Other schemes, like probabilistic road maps (Kavraki et al., 1996) and rapidly-exploring random trees (LaValle & Kuffner, 2001), instead generate a large set of discrete waypoints with links that connect them. Once discretization has occurred, both classes of techniques invoke heuristic search[1] to find a sequence of steps from the initial location to the target. However, the discretized space will typically include hundreds of cells or waypoints and require extensive search that humans do not appear to require.

Now consider another paradigm – feedback control – that emerged from the cybernetics movement – which emphasized analog representations (Bennett, 1996). This framework encodes states as points in a continuous space, with actions represented as numeric settings for control variables that change positions in this space. Calculation of control values is often influenced by measured differences between the current situation and a 'set point', which serves as a goal description. Although there clearly exists a space of possible paths through the state space, feedback control usually operates in a deterministic manner to produce a single trajectory without search. The classic approach is to provide one control equation for each effector (e.g., forward motion or turning angle), with 'proportional–integral–derivative' methods being a standard way to calculate control values. Continuous control excels at low-level tasks that require fine-grained, adaptive behavior, and it can be used for mental simulation of trajectories when a predictive model is available.

The basic framework of feedback control assumes that a single set of formulae determine agent behavior in a given setting. For situations that involve multiple objectives, a common variant in-

1. The favored method is often the A* algorithm, which Langley (2018) has argued subverts the original vision for heuristic search with its focus on finding optimal solutions despite the effort involved.

volves the use of potential fields (Khatib, 1985). These compute a set of attractive or repulsive forces, possibly including ones generated by waypoints, whose effects are summed and provided to control equations, leading to smooth behavior like that in Figure 1 (a). Another variant introduces a number of distinct 'modes', each with its own control equations, that operate under different conditions (Goebel, Sanfelice, & Teel, 2009). This discretizes the control process while still relying on continuous calculations within them. We will see shortly that the notion of modes maps well onto the abstract actions used in classic problem solvers, although we reiterate that our concern here lies with low-level motion planning rather than high-level task planning.

3. A Review of the PUG Architecture

We desire a computational theory that supports the embodied character of human cognition while retaining insights about high-level symbol processing. In a previous paper (Langley et al., 2016), we described PUG, a cognitive architecture for agents in continuous physical domains that:

- Grounds symbolic relations in quantitative descriptions of physical situations;
- Associates numeric utilities with symbolic goals that can reflect tradeoffs;
- Combines discrete actions with continuous parameters to specify fine-grained activities;
- Supports adaptive control that takes into account both the agent's objectives and its situation;
- Uses numeric simulation to generate motion trajectories that guide high-level planning.

PUG is a hybrid architecture that combines symbolic with numeric processing, elaborating on ideas from its predecessor ICARUS (Choi & Langley, 2018). In this section, we review the framework, examining commitments about its representation and cognitive processing, and its limitations.

3.1 Representation in the PUG Architecture

Like other cognitive architectures, PUG incorporates theoretical assumptions about the nature of mental content. The framework identifies three distinct types of long-term knowledge structures that remain stable over time. These include:

- *Concepts*, which encode generic symbolic relations among entities that are grounded in numeric attributes associated with those entities;
- *Motives*, which specify the conditions on when relations are desired or undesired, including their utility as a function of numeric attributes; and
- *Skills*, which encode generic discrete activities and their relational outcomes, but also include continuous equations for control attributes.

Concepts state how to decide what relations are true in the environment, motives indicate how to determine their value for the agent, and skills specify how to achieve them. Both unary categories (e.g., *tower*, *disk*) and relational concepts (e.g., *facing*, *between*) are defined in terms of these features. They offer one computational account of how cognition is *embodied*.

For example, an agent might include concepts for when it is *at* an object or *facing* it. These would be grounded in the agent's distances and angles relative to other entities in the environment. Its knowledge base might also contain motives that declare the desirability of being near a target

or far from an obstacle. These can refer to defined concepts, as well as to values of their numeric attributes. Moreover, the agent may have skills for approaching a target object or turning to face it, referring to defined concepts in its conditions or effects. These could incorporate equations for calculating the rate of forward motion or linear force and for angular motion or rotational force.

Furthermore, the PUG architecture postulates that these long-term structures have short-term, dynamic analogs, each of which is a specific instance of some corresponding generic element. These mental units include:

- *Beliefs*, which are concrete instances of concepts that specify symbolic relations and that have associated numeric attributes;
- *Goals*, which are desired or undesired beliefs with associated numeric utilities that can change as a function of the agent's situation; and
- *Intentions*, which are instances of skills that refer to specific entities and that predict beliefs that will result from their execution in the environment.

A collection of beliefs that the agent holds at the same time is a *state*, whereas a set of goals denotes a desired state that may or may not hold. Intentions are organized into sequential *plans*, which are further embedded in *search trees* that led to their construction.

3.2 Processing in the PUG Architecture

The PUG architecture also makes commitments about the mechanisms that operate over these mental structures. As in most computational treatments of extended cognition, these involve the repeated use of long-term content to update dynamic elements. The framework assumes three distinct processing levels, each with its own cognitive cycle:

- *Belief and state processing*, which matches concepts against percepts to infer beliefs, uses motives to compute goal utilities, and carries out a single step of skill execution.
- *Mental simulation*, which invokes state-level processing repeatedly to generate a trajectory of belief states, goals, and associated utilities.
- *Problem solving*, which carries out heuristic search through a space of task plans, each of which comprises a sequence of simulated intentions and associated utilities.

Each level builds on structures produced by the previous one, with state processing providing the raw material for mental simulation, which in turn offers the alternatives for problem solving and the evaluation scores to guide heuristic search.

State processing is related to the operation of production system architectures (Neches, Langley, & Klahr, 1987), as it relies on pattern matching and rule application to produce a sequence of mental states. Within this level, conceptual inference generates beliefs that lets the motivational module assign utilities to goals and that lets skill execution carry out agent intentions. However, it comes closer to Nilsson's (2001) design for a *tower architecture* and to ICARUS (Choi & Langley, 2018), which also separate inference from execution. Problem solving also mimics processing in found ICARUS, in that it searches for sequences of durative actions that achieve goals, but PUG relies on mental simulation of skills rather than on their execution in the environment.

3.3 PUG's Successes and Limitations

Langley et al. (2016) demonstrated PUG's abilities on ten mission scenarios that involved a robot delivering sensors to target sites in the presence of danger areas. They provided the system with skills for turning left or right to face an object, moving toward an object, refueling at depots, depositing sensors at targets, and collecting samples. These runs showed the architecture's ability to ground symbolic concepts in physical descriptions, combine discrete actions with continuous parameters, and invoke mental simulation to envision and evaluate trajectories, which it used to guide search for extended plans that reasoned about tradeoffs and inconsistent goals. In some cases, the system decided to abandon goals because side effects would offset the benefits of achieving them.

However, they also revealed important limitations, most of them at the level of continuous control. In particular, PUG could not carry out more than one skill at the same time, such as moving and turning toward an object in parallel, or modulate a skill's behavior to reflect other factors, such as encountering obstacles on the way to a target. For instance, it could generate the inefficient plan in Figure 1 (b) if provided with waypoints around obstacles but not the more natural trajectory in Figure 1 (a). Neither could the system consider the effects of natural processes, such as friction or headwinds, when predicting future states or consider alternative motion plans. An improved version of the architecture would address these issues while retaining the strengths of its predecessor.

4. A Theory of Motion Planning and Control

In response to these limitations, we have developed a revised theory of embodied agency that addresses them. In this section, we present tenets of the new framework that differ from the earlier one. As before, we first discuss postulates about representation and then turn to processes that operate over them. Readers can assume that unmentioned features of the architecture remain unchanged.

4.1 New Representational Postulates

The updated framework retains the original distinction among concepts, motives, and skills, but it alters them in important ways to support more flexible and fluid control.[2] One crucial change to the architectural theory is that it now posits:

- Concepts are *graded* in that they include functions to specify the degree to which they match. These refer to numeric attributes associated with entities or relations among them.

For instance, the match function for the concept *robot-at* might vary with the robot's distance to an object, whereas the one for *robot-facing* might refer to its angle. The idea that some entities or situations fit a concept description better than others is akin to the notion of *typicality* in accounts of categorization (Rosch & Mervis, 1975). This is not the same as incorporating probability distributions in the concept description, although it does not exclude them.

Another revision concerns skills, which encode knowledge about physical activities in which the agent can take part. In particular, the new framework postulates that:

2. Motives in the new theory encode the same content as in the earlier account, indicating the desirability or undesirability of relations in terms of numeric utility functions that are conditioned on the situation.

- Skills have associated *target concepts* they aim to achieve, as well as *control equations*, whose effects are proportional to the degree that these concepts are mismatched.

For example, a robotic agent's skill for moving to an object might include a *robot-at* relation as its objective, whereas one for turning to an object might have *robot-facing* as its target. Because such concepts match situations to varying degrees, they can serve as error signals, much like the difference between the state and a 'set point' in classic feedback control. We can also view such target concepts as sources for potential fields that attract or repel the agent. At the same time, they are analogous to symbolic effects that appear in operators for symbolic planning.

One drawback of the previous theory was that knowledge for both calculation of actions and their effects was embedded in skills, only one of which could be active at a time. This meant that the agent could not pursue multiple actions, such as moving forward and turning, in parallel, and it could not reason about external forces. In response, the new framework claims that:

- *Processes* specify how the attributes of objects in the environment change in response to their current values and the values of control attributes.

This revision factors knowledge about dynamics into skills, which determine the values of control attributes, and processes, which predict changes to the environment. It also lets the agent model the influence of mechanisms like wind and gravity over which it has no control.

Naturally, each of these assumptions about long-term cognitive structures have corresponding ones about shorter-term elements. These include the additional postulates that:

- Beliefs specify the *veracity* to which their concepts match and their *utility* obtained from motives;
- Intentions specify their *activation* based on mismatch of their target concept; and
- Events predict *changes* to attributes based on the current environment and control values.

As the agent or objects move in the environment, these quantities are updated, causing the numbers to vary while the basic relations remains the same. In AI planning research, a state is simply a conjunction of relations; in our framework, a state as a point in an N-dimensional space, where each dimension refers to a belief's veracity or utility or to an intention's activation level. The new theory also lets the agent pursue multiple skills in parallel, so it includes a new structure that specifies a *set* of intentions.[3] We will see that, when combined with other information, these imply a physical trajectory over time, so we will refer to such sets as *motion plans*. They play the same role in higher-level task plans as did single intentions in the earlier architecture.

4.2 New Processing Postulates

Of course, these extended representations can have no effect on agent behavior without augmented mechanisms to interpret them, which the new theoretical framework also incorporates. The first important change along these lines is that:

- Conceptual inference computes the *veracity* of each belief by evaluating the match function for the concept that it instantiates.

3. Clearly, some intentions are mutually exclusive, such as avoiding an obstacle on the left and the right, which suggests the need for another form of knowledge – *constraints* – that we will not address here.

This numeric score ranges from zero (no match) to one (a perfect match), with it varying over time in response to changes in the attributes used to compute it. For instance, the veracity of *(robot-at R1 O1)* will vary with the robot's distance from the object. Processing for motives is unchanged except that it assigns computed utilities directly to beliefs rather than to distinct goal structures.

The expanded theoretical framework also posits a more adaptive mechanism for interpreting skills that takes advantage of their refined structure and that is modulated by the results of graded conceptual inference:

- Skill execution treats the mismatch of each intention's target concept as its *activation*, which it combines with control equations to compute the values for control attributes.

This is a variation on feedback control that uses one minus the target concept's veracity as the error signal. Combined with the previous postulate, it lets the new theory retain the symbolic character of skills while gaining the flexible character of continuous control.

Another change to the framework is that an agent can carry out multiple intentions in parallel, on the same cognitive cycle, which naturally raises issues about how it handles interactions among them. Here the refined theory claims that:

- To determine the values for control attributes on a given cycle, skill execution sums the results computed by different intentions that affect them.

For instance, an intention for approaching object *O1* may cause a turn to the right, while another for avoiding object *O2* may cause a turn to the left. In such cases, the influences on control attributes are added, much as in the calculation of vector sums in potential field approaches to control (Khatib, 1985). In addition, the framework includes mechanisms for using processes to make predictions:

- Process interpretation computes expected changes to values for state attributes as a function of current values for state attributes and control attributes.

As with skill execution, if multiple processes affect the same attribute, then their influences are combined. For example, the robot's forward motion resulting from control settings will be offset if it encounters a headwind, with the resultant movement being their vector sum.

Finally, the possibility of parallel skill execution introduces a new level of choice for agents, since they must decide which set of intensions to pursue. Each such set, if executed, will produce a distinct trajectory over time and, combined with process-driven prediction, the agent can simulate these trajectories to select among them. We will refer to search through the space of intention sets as *motion planning*, a cognitive activity that occupies an intermediate level between state processing and task planning that did not occur in the previous framework.

5. The PUG/C Architecture

We have incorporated these theoretical ideas into PUG/C, an extension of the previous PUG architecture (Langley et al., 2016). This earlier system also relied on grounded concepts and embedded continuous behavior within symbolic skills, but it did not support graded categories, feedback control, parallel uses of intentions, process-guided prediction, or motion planning. In this section we describe the new architecture, focusing first on its representation of cognitive structures and then on how it interprets them to achieve an agent's objectives in an environment.

Table 1. (a) Two PUG/C concepts for the robot domain, each of which includes a head, observed elements, and a veracity function based on the entities' attributes. The piecewise linear function $(linear\ obs\ max\ min)$ returns 1.0 if the observed value $obs \leq max$, returns 0.0 if $obs \geq min$, and returns $obs/(max - min)$ when $max < obs < min$. (b) Two PUG/C motives for the same domain, which specify utility functions for beliefs in their heads and indicate whether they address achievement or maintenance goals.

```
(a) ((robot-at ^id (?r ?o) ^distance ?d)
      :elements   ((robot ^id ?r ^radius ?rr)
                   (object ^id ?o ^distance ?d ^radius ?or))
      :veracity   ((linear ?d (+ ?rr ?or) 10.0)))
    ((robot-facing ^id (?r ?o) ^angle ?a)
      :elements   ((robot ^id ?r) (object ^id ?o ^angle ?a))
      :veracity   ((linear ?a 0.0 45.0)))
```

```
(b) ((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)
    ((approaching ^id (?r ?o))
      :conditions ((robot ^id ?r ^radius ?rr)
                   (object ^id ?o ^type obstacle ^distance ?d ^radius ?or))
      :function   (cond ((< ?d (+ ?rr ?or)) -20.0) (t 0.0))
      :type       maintenance)
```

5.1 PUG/C Representation and Syntax

PUG/C provides a programming language that supports the construction of embodied intelligent agents. The syntax covers long-term knowledge structures that are stable over time and short-term elements that change during processing. This extends the earlier PUG notation that reflect its new theoretical commitments. Table 1 (a) shows two *conceptual rules* for the two-dimensional robotic domain depicted in Figure 1. These define the relations *robot-at* and *robot-facing*. Each rule has a head that specifies a predicate and a set of attribute values, including an identifier for the relation. In addition, an *:elements* field describes a set of typed objects, each with an identifier and numeric attribute values, and an optional *:tests* field with Boolean tests that must be satisfied for the concept to match. The *:veracity* field specifies how to compute the concept's degree of match as a function of bound variables; this supports the notion of graded category membership.

In addition, Table 1 (b) provides two motives from the robotics domain that specify utility functions for the relations *robot-at* and *approaching*. These functions refer to variables bound in the conditions, such as the robot's radius *?rr*, the object's radius *?or*, and the robot's distance *?d* to the object. Because some matched values will change over time, the computed score will vary in turn. The first structure is an achievement motive, which generates utility only when it first matches for a particular belief. The second is a maintenance motive, which assigns utility to a belief repeatedly on every cycle for which its conditions match. The latter type often specifies negative values, which means that the agent should avoid them if possible.

Table 2. Four percepts for the robot domain that describe objects the agent perceives for the scenario in Figure 1 (a) and five beliefs that describe its relations to these objects. Each structure includes a predicate, an identifier (which may be a list), attribute values, and a veracity score (in brackets) between zero and one. Beliefs that have a score below 0.5, such as *(robot-at ^id (R1 O3)*, do not appear in memory.

```
(robot ^id R1 ^radius 0.15 ^move-rate 0.0 ^turn-rate 0.0) [1.0]
(object ^id O1 ^distance 2.0 ^angle 0.0 ^radius 0.4) [1.0]
(object ^id O2 ^distance 4.123 ^angle 14.032 ^radius 0.4) [1.0]
(object ^id O3 ^distance 6.0 ^angle 0.0 ^radius 0.4) [1.0]
(robot-at ^id (R1 O1) ^distance 2.0) [0.847]
(robot-at ^id (R1 O2) ^distance 4.123) [0.622]
(robot-facing ^id (R1 O1) ^angle 0.0) [1.0]
(robot-facing ^id (R1 O2) ^angle 14.032) [0.688]
(robot-facing ^id (R1 O3) ^angle 0.0) [1.0]
```

Table 2 presents examples of *beliefs* that describe the scenario in Figure 1 from the robot's egocentric perspective. The four percepts – primitive beliefs that come directly from the environment or prediction – each specify an object type, an object identifier, and attribute values that describe it. The latter include the object's distance and angle from the robot in agent-centered polar coordinates, along with their radii, since they have circular cross sections. The table also contains five inferred beliefs (instances of defined concepts) about the robot's relations – *robot-at* and *robot-facing* – to the perceived objects *O1*, *O2*, and *O3*. These contain more than symbolic predicates; they include attribute values derived from constituent entities. Moreover, each belief includes a *:veracity* score that reflects how well it matches the respective concept. For instance, *(robot-facing R1 O2)* has the score 0.688 because the veracity expression *(linear 14.032 0.0 45.0) = 0.688*. This score, along with the belief's derived attribute values, can change over time while its symbolic aspects remain stable.

PUG/C also provides a syntax for skills that describe how to achieve the agent's objectives. Table 3 (a) presents two examples from the robot domain, one for moving toward a given object and another for turning to face an object. Each skill has a head that specifies a name and set of arguments, along with an *:elements* field that describes the arguments' types and values for a subset of their attributes. As in conceptual rules, a *:tests* field includes Boolean tests that must be satisfied for the skill to apply. Most important, a skill specifies a *:target* relation that it aims to achieve and a *:control* field with expressions for computing values for control attributes as a function of the degree to which the target concept is *mismatched* (i.e., one minus the target belief's *:veracity* score).

The architecture also incorporates a notation for processes, with two examples shown in Table 3 (b). This includes an *:elements* field that describes arguments and types, along with an optional *:tests* field with Boolean expressions. The key differences from skills are that they specify no control equations and they lack a target concept, as they are not teleological in character. Instead, they incorporate a *:changes* field that specifies how values for attributes of one or more entities will change as a function of variables matched in the *:elements* field. Thus, they encode causal knowledge about the effects of actions or environmental forces.

Just as beliefs in PUG/C are instances of general concepts, so *intentions* are instances of skills and *events* are instances of processes. Each intention provides a skill name and arguments, along

Table 3. (a) Two skills for the robot domain, each of which includes a relational head, a set of observed entities, arithmetic tests, one or more control equations, and a target concept. The *move-to* skill influences the control attribute *move-rate*, while *turn-to* affects *turn-rate*. (b) Two processes that describe the effects of these control attributes on the robot's *distance* and *angle* to an object. The symbols $\star da$ and $\star da$ denote trigonometric functions for computing linear and angular change.

```
(a)((move-to ?r ?o)
     :elements ((robot ^id ?r ^turn-rate ?t) (object ^id ?o ^angle ?a))
     :tests    ((> ?a -90) (< ?a 90))
     :control  ((robot ^id ?r ^move-rate (* 0.3 $MISMATCH)))
     :target   ((robot-at ^id (?r ?o))))
   ((turn-to ?r ?o)
     :elements ((robot ^id ?r)
                (object ^id ?o ^angle ?a ^distance ?d))
     :control  ((robot ^id ?r ^turn-rate (* 5.0 (sign ?a) $MISMATCH)))
     :target   ((robot-facing ^id (?r ?o))))
```

```
(b)((move-relative ?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))))
   ((turn-relative ?r ?o)
     :elements ((robot ^id ?r ^turn-rate ?t)
                (object ^id ?o ^angle ?a))
     :changes  ((object ^id ?o ^angle (* -1.0 ?t))))
```

with a mismatch score for the target belief and values for associated control attributes. For instance, for the scenario in Figure 1 (a), the agent might have the intention set *((move-to R1 O3) (turn-to R1 O3) (avoid-on-left R1 O1) (avoid-on-right R1 O2))*. Each intention's target mismatch and control values would vary over time, while its symbolic description would remain the same. Events encode only the process name, its arguments, and the derivatives they predict for the current situation.

Finally, PUG/C incorporates the theory's notion of encoding *motion plans* as sets of intentions. The continuous trajectories associated with these structures are generated as needed, during mental simulation, rather than stored in memory. However, the intention sets are retained either as basic steps in higher-level task plans, which are themselves embedded in search trees that contain alternative courses of action considered by the agent during problem solving.

5.2 Belief and State Processing

Like other cognitive architectures, PUG/C operates in discrete cycles, but this occurs at five cascaded levels of processing, with results from each one providing raw material for the next. The two lowest levels focus on belief and state processing, with the first addressing conceptual inference. On each inference cycle, the module finds all conceptual rules with elements whose types match against current percepts. For each match, the system computes derived attribute values, calculates the *:veracity* score, and, if this exceeds a threshold, adds an inferred belief to a state memory. Next the module finds rules with elements whose types match a subset of the percepts and these inferred

beliefs, leading to new beliefs that are added to the state description if their scores are high enough. This procedure continues until no more conceptual matches occur, giving a set of beliefs and percepts that encode the current state. For instance, given the scenario in Figure 1, the concepts in Table 1 and the percepts in Table 2 would produce the beliefs in Table 2.

At the second level, PUG/C's state-processing module examines the current set of intentions. In each case, it checks to see whether the elements and tests are satisfied by the current belief state. If so, then the module accesses each intention I's target belief and its associated *:veracity* score. The interpreter substitutes one minus this score for the symbol *$MISMATCH* in I's control equations, along with the values for any bound variables. Next the system evaluates the instantiated expression to compute the value for each control attribute. For example, given the intention *(move-to R1 O1)* and the *:veracity* score 0.518 for target belief *(robot-at ^id (R1 O1))*, the mismatch would be $1 - 0.518 = 0.482$ and the *move-rate* would be $0.3 \times 0.482 = 0.145$. When multiple intentions apply on a cycle, the module computes target mismatches and control values for each one. If different intentions affect a control attribute, then PUG/C takes the vector sum of their outputs.

Once the architecture has computed the summed values for its control attributes, it uses matched instances of processes (events) to predict changes to the environment. Some processes calculate the effects of agent actions, whereas others describe only external influences. Each event predicts a change to one or more environmental attributes, with their individual effects being summed to determine overall results. For instance, given the value 0.145 for the control attribute *move-rate* from above, the current *distance* of 5.105 and *angle* of 33.189 to *O1*, the event *(move-relative R1 O1)* predicts a change of -0.121 for *distance* and 0.913 for *angle*. Different instances of the *move-relative* process also predict changes to the robot's *distance* and *angle* to other objects.

The state-processing module also matches all motives against the current beliefs. For each matched instance, it substitutes bound variables into the associated utility function, computes the result, and deposits it in the *:utility* field of the corresponding belief. If multiple motives have the same head, PUG/C assigns the sum of their utilities to this belief. This mechanism is identical to that in the original architecture except that it creates no separate goal structures. Instead, it stores utilities with beliefs themselves, even when their *:veracity* scores are too low to include them otherwise. In such cases, they encode desired beliefs that do not yet hold and play the role of unsatisfied goals.

5.3 Mental Simulation and Motion Planning

The next two layers are responsible for mental simulation and motion planning. The first relies on a simple iterative mechanism. Recall that, combined with the calculation of control attributes by skill execution, processes let the architecture predict the next physical state in terms of what the agent can expect to perceive. This lets it apply conceptual inference to create a new belief state, which in turn lets it invoke skill execution, process prediction, and utility calculation. When applied repeatedly, this mechanism of mental simulation produces a trajectory of belief states over time. Each simulated trajectory follows deterministically from an initial situation and a set of intentions.

PUG/C users can provide such a motion plan manually, but it can also find such structures through heuristic search, using mental simulation and utilities to guide it. This planning mechanism starts with some desired relations, which PUG/C uses to retrieve candidate intentions by matching against skills' target concepts, much as in backward-chaining planners. If some relations produce

multiple intentions, then the system considers all combinations, as there will seldom be many, with each set of intentions serving as a motion plan from which to initiate search. The architecture uses mental simulation to envision each plan's trajectory if it were executed in the environment. On each time step, it uses motives to compute each beliefs' utility and stores a running total of values associated with them. If some of these beliefs are sources of negative utility, then PUG/C retrieves new intentions that should reduce or eliminate them, favoring ones that occurred earlier in the trajectory based on stored timing statistics. The system adds each intention to the current motion plan, simulates each elaboration, and selects the best one. However, the new trajectory may introduce further sources of negative utility (e.g., obstacles) that require additional repairs. This process continues until it finds no way to improve on the best candidate, at which point it halts.

In summary, PUG/C carries out greedy search through a space of motion plans, starting with initial intentions that address target beliefs and elaborating them to improve average utility. Mental simulation produces a continuous trajectory for each node in the search tree, but every candidate comprises a discrete set of intentions. This approach differs from classic methods, which either introduce waypoints that guide continuous control or search through a fine-grained discretized space. In contrast, PUG/C combines conceptual inference, feedback control, and process-driven prediction to generate deterministic trajectories, then uses motives to assign utilities and to retrieve intentions that improve on them. The architecture passes the selected motion plans to its problem solver, which uses them as elements in a higher-level search for task plans and makes decisions about conflicting top-level goals. This facet has not changed from the earlier version of the PUG architecture.

At first glance, it seems simpler to generate all possible intentions, say by matching skills against perceived objects, and then combine them into a single motion plan. This scheme appears tractable even for environments with 20 or 30 entities, as mental simulation is a deterministic process for a given set of intentions. However, it is impractical because some intentions, such as bypassing an obstacle *O1* on both the left and the right, would cancel each other out, so a plan that includes both would not avoid *O1* after all. We could use domain constraints to ensure that a motion plan includes only nonconflicting intentions, but a scenario with N obstacles would then produce 2^N motion plans, each requiring simulation, with the number growing exponentially. Humans rely on heuristic search rather than exhaustive techniques in all but the simplest cases, which recommends the greedy method described above, including use of negative sources of utility to propose intentions.

6. Empirical Demonstrations

To demonstrate that PUG/C supports the target abilities described earlier, we developed a two-dimensional environment in which a mobile robot can sense its surroundings and control its motion. The agent perceives other nearby objects, including attributes like their radii, distances, and angles. The robot's absolute position and orientation is known to the simulator but not to the agent itself, which senses relations to other entities in egocentric polar coordinates. The environment obeys a form of Aristotelian physics, in that objects move only in response to explicit actions or forces. The agent can set values for two control attributes – *move-rate* (distance per cycle) and *turn-rate* (angles per cycle). We can create different scenarios by specifying the robot's initial pose and the number, type, location, and attributes of other objects. We can also include natural processes that affect the robot or other objects, leading to nonvolitional changes.

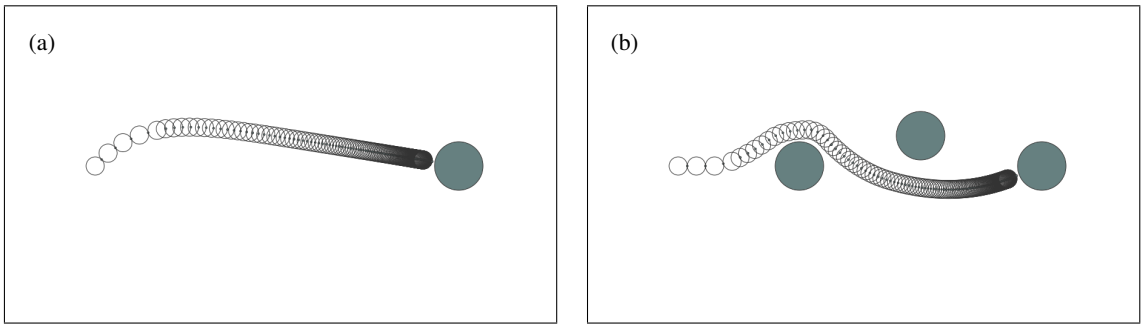

Figure 2. Two scenarios in which a robot must approach a nearby object: (a) when the robot is facing away from the target, it turns and moves forward simultaneously; (b) when two obstacles fall on the robot's path to the target, it veers left to avoid the first and right to evade the second. Traces show the robot's pose at equal time intervals, illustrating that its position and orientation change more slowly as it approaches the objective.

6.1 Combining Intentions with Processes to Simulate Trajectories

Our first demonstrations show that PUG/C can carry out multiple intentions in parallel and use processes to generate a simulated trajectory of environmental behavior. We provided the system with two concepts *robot-at* and *robot-facing* from Table 1, along with the skills *move-to* and *turn-to* from Table 3. We also gave it two intentions (instances of these skills) – *(move-to R1 O1)* and *(turn-to R1 O1)* – with target beliefs *(robot-at R1 O1)* and *(robot-facing R1 O1)*, respectively. Figure 2 (a) shows the behavior produced by these knowledge structures when *O1* is initially 6.0 units away from the robot and 45 degrees to its right. Conceptual inference shows that *(robot-facing R1 O1)* has an imperfect match score (initially 0.0), so the system invokes the intention *(turn-to R1 O1)*, which sets a positive value the control attribute *turn-rate* (initially -10.0). The intention *(move-to R1 O1)* also has a poorly matching target belief, *(robot-at R1 O1)*, so it sets the value for the control attribute *move-rate*. These lead the robot's position and orientation to change in parallel, with both slowing down over time as the *:veracity* score for each target belief approaches unity. The run continues for 145 iterations, but *(turn-to R1 O1)* becomes inactive after cycle 137, as the robot is facing *O1* at this point and it needs only to move forward.

We also tested PUG/C on scenarios that require the robot to avoid obstacles it encounters on the way to a target object. Here we provided the system with the same concepts and skills as in the previous run. However, we included a new concept, *approaching*, that holds when the robot is approaching an object, with the veracity depending on how much it will clear the obstacle given its current course. In addition, we gave it four intentions – *(move-to R1 O3)*, *(turn-to R1 O3)*, *(avoid-on-left R1 O1)*, and *(avoid-on-right R1 O2)*. The latter referred to two skills for avoiding objects on the left and on the right, both with *approaching* as a negated target concept. These do not affect the *move-rate* parameter; they only alter the robot's turn rate, which is necessary to swerve around the obstruction. This influence conflicts directly with the agent's intention to face the target object, but its effect becomes stronger as the robot gets closer to the obstacle and halts after it has passed. In the meantime, the intention for moving to the target continues to draw the robot forward.

Figure 2 (b) shows the resulting behavior. In this run, the intention *(avoid-on-left R1 O1)* only starts to affect behavior on cycle 3, when the robot first approaches object *O1*. At this point, its effect on *turn-rate* becomes dominant and the robot starts to veer around the obstacle. However, once it has passed *O1*, this influence disappears and the robot turns back to *O3* until it comes close to *O2*, the obstacle. Starting on cycle 14, *(avoid-on-left R1 O2)* causes the robot to turn in the other direction, but its influence ends by cycle 18, when collision is no longer imminent. Note that *(turn-to R1 O3)* is active during most of the run, but the avoidance intentions have greater impact. These traces provide evidence that the architecture combines intentions in an effective manner, reproducing the behavior of potential fields in classic control systems. They also show it can combine skill execution with process prediction to mentally simulate motion plans that involve continuous behavior.

6.2 Searching for Motion Plans

We also wanted to demonstrate PUG/C's ability to generate motion plans using heuristic search. We presented the system with the scenario in Figure 3 (a), in which the robot on the left must reach the target object on the right while avoiding three obstacles. The architecture initially retrieves two intentions, *(move-to R1 O3)* and *(turn-to R1 O3)*, that should achieve the two target beliefs we provided, *(robot-at R1 O3)* and *(robot-facing R1 O3)*. This produces the trajectory in Figure 3 (b), which achieves these objectives but has the average utility -1.137 because it runs through object *O1*. The belief *(approaching R1 O1)* is the only source of negative utility, so PUG/C retrieves two intentions – *(avoid-on-left R1 O1)* and *(avoid-on-right R1 O1)* – either of which should eliminate it. The system adds each one to the original set of intentions and simulates both plans, producing the trajectories depicted in Figure 3 (c) and (d).

Both motion plans have average utilities of -0.652 because, although they bypass *O1*, their trajectories intersect another obstacle, one on the left and another on the right. Because they have equal scores, PUG/C selects one of them at random, say the option in (c), and examines the source of negative utility, in this case *(approaching R1 O2)*. As before, the system retrieves two intentions that could eliminate this relation: *(avoid-on-left R1 O2)* and *(avoid-on-left R1 O2)*. The architecture adds each option to a new plan and simulates their behavior to generate the trajectories in Figure 3 (e) and (f). The first alternative, in which the robot turns right, has an average utility of 0.097, whereas the second, in which it turns left another time, has 0.090 as its utility. For this reason, PUG/C selects the former and, since it involves no other sources of negative value, returns this motion plan as the final result. The module's greedy approach is not guaranteed to find the most efficient path, but it usually finds a reasonable one with little search effort.

7. Related Research

The approach to motion planning and continuous control just described resembles some of its predecessors, but there are important disparities. Our primary objective has been theoretical: to extend ideas about cognitive architectures, which focused originally on symbolic reasoning and problem solving, to support adaptive and smooth behavior in continuous domains. For this reason, we need not address whether our framework is superior to 'state of the art' methods, as they were developed with different goals in mind. However, we can compare the new theory, and its implementation in PUG/C, to earlier work in the same general area to identify key similarities and differences.

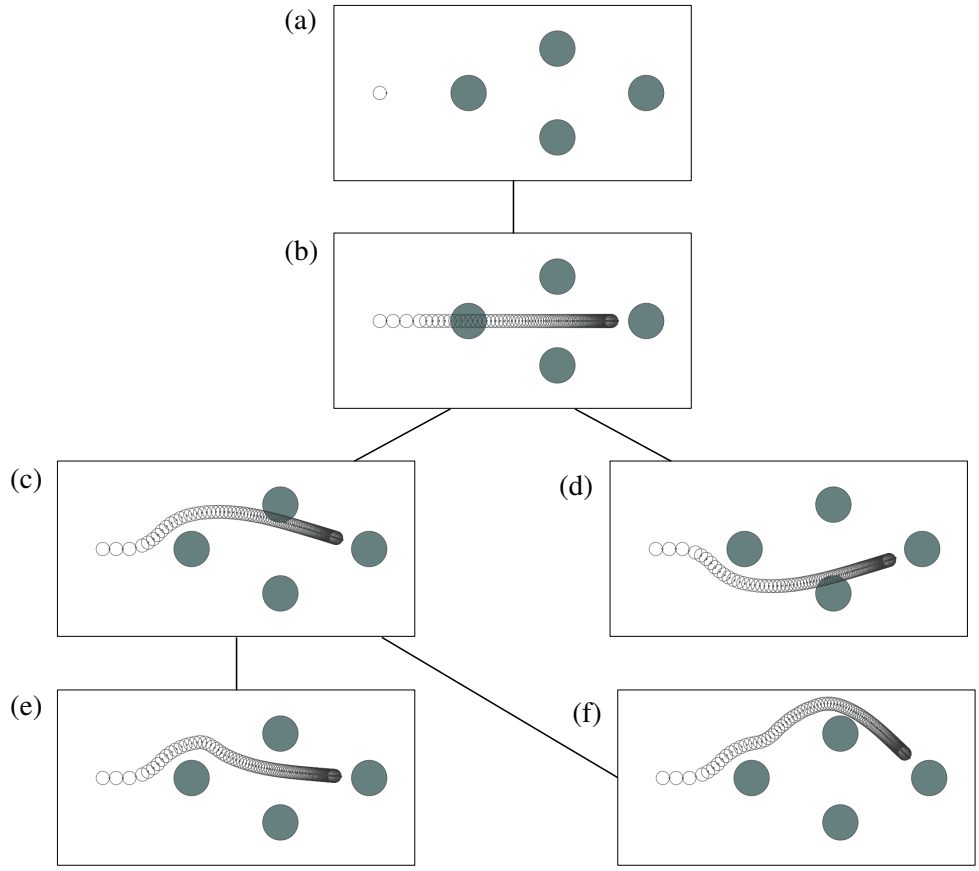

Figure 3. The tree explored during heuristic search for a motion plan that will take the robot from its initial position on the far left to the target object on the far right. At each level of search, PUG/C adds candidate intentions that should improve utility, generates a simulated trajectory, and selects the best alternative based of scores. In this run, the system settles on the motion plan depicted in (e).

The most closely related efforts involve other extensions to cognitive architectures and symbolic planners. For instance, variants of Soar (Laird, 2012) and ACT-R (Salvucci, 2006) incorporate feedback control for physical settings, but they invoke it as a subroutine within cognitive cycles, rather than bringing it fully into the architectures. This is similar in spirit to 'hybrid' control systems (Goebel, Sanfelice, & Teel, 2009), which embed continuous processing within discrete finite-state networks, with different equations associated with each state or mode. The ICARUS architecture (Choi & Langley, 2018) includes durative skills that have continuous effects, but it cannot use them to plan smooth motion trajectories. Extensions to planning formalisms like PDDL+ (Cashmore et al., 2020; Fox & Long, 2006) support similar representations, in that they augment symbolic operators with difference equations that update continuous attributes, but they do not support error-driven feedback control. Moreover, research along these lines has emphasized task-level planning, whereas we have focused instead on motion planning.

There is also a substantial literature on integrated task and motion planning (Garrettt et al., 2021). However, this relies on traditional techniques for each level and instead concentrates on how to combine them. The primary alternatives are (1) first identifying promising waypoints in continuous space and then finding discrete plans that connect them, (2) first searching for a discrete task plan and then finding a motion plan for each step, and (3) interleaving the two processes so they mutually constrain each other's search. Our approach to motion planning lends itself to last two schemes, which suggests ways to integrate it with PUG's task-level problem solver (Langley et al., 2016). Nevertheless, we should note again that we have addressed here only the generation of parallel motion plans, not the construction of sequential task plans.

As we discussed in Section 2, classic techniques for motion and path planning address continuous settings in a very different manner. They discretize the environment by dividing them into grid cells or introducing waypoints that are linked into graphs, as in methods for probabilistic road maps (Kavraki et al., 1996) and rapidly-exploring random trees (LaValle & Kuffner, 2001). After this, they invoke heuristic search to find a path from an initial position or pose to the target, transforming the problem of continuous motion planning into a problem of discrete search. However, these solutions seem unnecessarily complicated and run counter to intuitions about the amount of search that humans require to generate movement trajectories. Our framework involves some limited search, but only in that it must decide which intentions to include, as these determine a trajectory through space and time. One can design scenarios that lure this approach into local optima and fail to achieve the target (e.g., with *cul de sacs*), but these are atypical.

We should also consider research on qualitative approaches to robot planning and control, which rely on similar representational assumptions. For instance, Brenner et al. (2007) report a system to place objects in positions that satisfy qualitative spatial relations and even draws on potential fields to guide behavior. Wiley et al. (2016) present an architecture for mobile robots that uses a qualitative formalism, based on landmark values, to specify states, control attributes, and learned conditions for the latter's values. Hasan et al. (2020) describe a system that represents a cluttered environment as a coarse grid through which a robot gripper must maneuver to reach a target using learned qualitative heuristics. However, all three efforts resort to multi-step planning that generate discrete action sequences rather than the continuous motion planning that PUG/C employs. In contrast, our theory comes much closer to Kuipers' (2000) spatial semantic hierarchy, which associates quantitative control laws with qualitative regions that drive a mobile robot to distinctive states. One important difference is its emphasis on cognitive maps, a topic that we have only started to explore.

Another core tenet of our theory is its combination of inference with control, an area that has also been widely studied. However, most previous work has focused on estimating the probabilities for simple state encodings from noisy or incomplete sensor readings, rather than creating rich relational descriptions. Cognitive architectures like Soar and ICARUS provide explicit mechanisms for elaboration of sensed percepts to form higher-level beliefs, but they do not easily support graded category membership. Choi's (2010) extension to ICARUS incorporated this notion, but his variant did not use the degree of conceptual match as an error signal to produce adaptive behavior. Fuzzy controllers (Katz, 1997; Zadeh, 1968) employ membership functions in much the same way as PUG/C but they typically encode only low-level state descriptions. Instead, our framework augments symbolic inference of relational descriptions with a veracity score to provide continuous feedback.

A final theoretical assumption is that sequential action is guided by differences between the current and desired state. Of course, continuous control methods rely on error signals to calculate values for control attributes, but the notion of reducing differences was also central to operation of the General Problem Solver (Newell et al., 1960), arguably the first symbolic planner. Nilsson's (1994, 2001) teleoreactive framework uses symbolic control rules to bring embodied agents closer to desired ends in continuous environments, but it did not rely on numeric error. ICARUS uses means-ends analysis to direct choices about which actions to carry out, but only at the symbolic level. Our framework is distinctive in that it uses degree of match for symbolic goals (target concepts) to drive continuous control within the context of discrete skills. The target concepts associated with skills also serve as sources for potential fields that combine to balance the agent's competing objectives.

In closing, we should remind the reader that research on cognitive architectures aims to develop unified theories of the mind. Thus, it is not appropriate to ask whether any particular component of our theory, or its implementation in PUG/C, is superior to alternative approaches for the same subproblem. A more reasonable question is whether these components fit together in elegant ways that support each other and whether they produce human-like behavior. One can also ask whether the new elements augment the preceding version of the architecture in promising ways that extend the range of behaviors it covers. We maintain that our new account of continuous control, mental simulation, and motion planning fares well on these criteria.

8. Concluding Remarks

In this paper, we presented extensions to a theory of embodied intelligence that combines ideas from symbolic reasoning and continuous control. We motivated the research by a desire to extend PUG, an existing cognitive architecture, to support smooth trajectories that approach targets while avoiding obstacles. We described the theory's postulates about mental structures and the cognitive mechanisms that operate on them. These assume that inference matches concepts to different degrees, skill execution use these match scores to calculate control values, processes support mental simulation of multiple intentions, and heuristic search finds high-utility motion plans. In addition, we described PUG/C, an augmented architecture that provides a syntax for encoding concepts, motives, skills, and processes, along with an interpreter that supports these target abilities. We demonstrated these abilities in a simulated robotics environment, including scenarios that required parallel skill application, fluid obstacle avoidance, and generation of motion plans.

The initial evidence suggests that the extended theory has substantial promise, but it is also clear that considerable work remains. Future research should add new abilities to:

- Specify *places* – virtual objects – in terms of their distances to reference objects and to use their inferred distances and angles during inference, control, and planning;
- Incorporate these virtual objects into other place definitions to encode topological networks that characterize large-scale *maps* and use these connections to guide path planning;
- Recover from local optima reached during motion planning (e.g., *cul de sacs*) by creating virtual objects, with associated motives, that serve as attractive or repulsive waypoints;
- Represent and reason about objects with complex boundaries, and relations among them, in terms of both qualitative relations and quantitative parameters.

We have reported preliminary results on the first two topics (Langley & Katz, 2022), but spatial cognition involves a complex set of abilities and has a rich literature in both psychology and robotics, while our initial forays have only scratched its surface.

In addition, we should demonstrate that PUG's existing mechanisms for task planning (Langley et al., 2016) interacts as intended with the new module for generating motion plans. Similarly, we should show the same holds for an earlier extension that interleaves task planning with anomaly-driven execution monitoring (Langley et al., 2017). Finally, we should test the new framework on complex scenarios in richer domains, such as urban driving, that require reasoning about tradeoffs among competing objectives during planning and execution (Langley, 2019). Together, progress on these fronts will produce an even stronger architectural theory of embodied intelligent agents.

Acknowledgements

The research reported here was supported by Grant No. FA9550-20-1-0130 from the US Air Force Office of Scientific Research, which is not responsible for its contents. We thank Mohan Sridharan and the reviewers for constructive comments that improved the paper considerably.

References

Bennett, S. (1996). A brief history of automatic control. *IEEE Control Systems Magazine*, *16*, 17–25.

Brenner, M., Hawes, N., Kelleher, J., & Wyatt, J. (2007). Mediating between qualitative and quantitative representations for task-orientated human-robot interaction. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (pp. 2072–2077). Hyderabad, India.

Cashmore, M., Magazzeni, D., & Zehtabi, P. (2020). Planning for hybrid systems via Satisfiability Modulo Theories. *Journal of Artificial Intelligence Research*, *67*, 235–228

Choi, D. (2010). *Coordinated execution and goal management in a reactive cognitive architecture*. Doctoral dissertation, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA.

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

Fox, M., & Long, D. (2006). Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research*, *27*, 235–297.

Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., & Lozano-Perez, T. (2021). Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, *4*, 265–293.

Goebel, R., Sanfelice, R. G., & Teel, A. R. (2009). Hybrid dynamical systems. *IEEE Control Systems Magazine*, *29*, 28–93.

Hasan, M., Warburton, M., Agboh, W. C., et al. (2020). Human-like planning for reaching in cluttered environments. *Proceedings of the 2020 IEEE International Conference on Robotics and Automation* (pp. 7784–7790). Paris: IEEE Press.

Katz, E. P. (1997). Extending the teleo-reactive paradigm for robotic agent task control using Zadehan (fuzzy) logic. *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation* (pp. 282–286). Monterey, CA.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. *Proceedings of the 1985 IEEE International Conference on Robotics and Automation* (pp. 500–550). St. Louis.

Kuipers, B. (2020). The Spatial Semantic Hierarchy. *Artificial Intelligence*, *1–2*, 191–233.

Laird, J. E., Kinkade, K. R., Mohan, S., & Xu, J. Z. (2012). Cognitive robotics using the Soar cognitive architecture. *Proceedings of the AAAI-2012 Cognitive Robotics Workshop*. Toronto.

Langley, P. (2018). Planning systems and human problem solving. *Advances in Cognitive Systems*, *7*, 13–22.

Langley, P. (2019). Explainable, normative, and justified agency. *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence* (pp. 9775–9779). Honolulu, HI: AAAI Press.

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). Extending an embodied cognitive architecture with spatial representation and reasoning. *Proceedings of the Third International Workshop on Human-Like Computing*. Windsor Great Park, UK.

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

LaValle, S. M., & Kuffner, J. J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, *20*, 378–400.

Neches, R., Langley, P., & Klahr, D. (1987). Learning, development, and production systems. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press.

Newell, A., Shaw, J. C., & Simon, H. A. (1960). Report on a general problem-solving program for a computer. *Proceedings of the International Conference on Information Processing* (pp. 256–264). UNESCO House, France: UNESCO.

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

Nilsson, N. (2001). Teleo-reactive programs and the triple-tower architecture. *Electronic Transactions on Artificial Intelligence*, *5*, 99–110.

Rosch, E., & Mervis, C. B. (1975). Family resemblance studies in the internal structure of categories. *Cognitive Psychology*, *7*, 573–605.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *48*, 362–380.

Selz, O. (1922). *On the psychology of productive thinking and of error*. Bonn, Germany: Cohen.

Wiley, T., Sammut, C., Hengst, B., & Bratko, I. (2016). A planning and learning hierarchy using qualitative reasoning for the on-line acquisition of robotic behaviors. *Advances in Cognitive Systems*, *4*, 93–112.

Zadeh, L. A. (1968). Fuzzy algorithms. *Information and Control*, *12*, 94–102.